A MARSHALL CAVENDISH **51** COMPUTER COURSE IN WEEKLY PARTS

INPUT

LEARN PROGRAMMING - FOR FUN AND THE FUTURE



UK £1.00
Republic of Ireland £1.25
Malta 85c
Australia $2.25
New Zealand $2.95

INPUT

Vol. 4 No 51

INDEX

The last part of INPUT, Part 52, will contain a complete, cross-referenced index. For easy access to your growing collection, a cumulative index to the contents of each issue is contained on the inside back cover.

PICTURE CREDITS
Front cover, Graeme Harris. Pages 1589, 1590, 1591, 1593, 1594, 1595, Anni Axworthy. Pages 1597 to 1618, Graeme Harris. Pages 1618, 1619, Berry Fallon Design.

Published by Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA, England. Typeset by MS Filmsetting Limited, Frome, Somerset. Printed by Cooper Clegg Web Offset Ltd, Gloucester and Howard Hunt Litho, London.



There are four binders each holding 13 issues.

HOW TO ORDER YOUR BINDERS

UK and Republic of Ireland:
Send £4.95 (inc p & p) (IR£5.95) for each binder to the address below:
Marshall Cavendish Services Ltd, Department 980, Newtown Road, Hove, Sussex BN3 7DN
Australia: See inserts for details, or write to INPUT, Times Consultants, PO Box 213, Alexandria, NSW 2015
New Zealand: See inserts for details, or write to INPUT, Gordon and Gotch (NZ) Ltd, PO Box 1595, Wellington
Malta: Binders are available from local newsagents.

BACK NUMBERS

Back numbers are supplied at the regular cover price (subject to availability).
UK and Republic of Ireland:
INPUT, Dept AN, Marshall Cavendish Services, Newtown Road, Hove BN3 7DN
Australia, New Zealand and Malta:
Back numbers are available through your local newsagent.

COPIES BY POST

Our Subscription Department can supply copies to any UK address regularly at £1.00 each. For example the cost of 26 issues is £26.00; for any other quantity simply multiply the number of issues required by £1.00. Send your order, with payment to:
Subscription Department, Marshall Cavendish Services Ltd, Newtown Road, Hove, Sussex BN3 7DN
Please state the title of the publication and the part from which you wish to start.

HOW TO PAY: Readers in UK and Republic of Ireland: All cheques or postal orders for binders, back numbers and copies by post should be made payable to:
Marshall Cavendish Partworks Ltd.

QUERIES: When writing in, please give the make and model of your computer, as well as the Part No., page and line where the program is rejected or where it does not work. We can only answer specific queries – and please do not telephone. Send your queries to INPUT Queries, Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA.

INPUT IS SPECIALLY DESIGNED FOR:

The SINCLAIR ZX SPECTRUM (16K, 48K, 128 and +), COMMODORE 64 and 128, ACORN ELECTRON, BBC B and B+, and the DRAGON 32 and 64.

In addition, many of the programs and explanations are also suitable for the SINCLAIR ZX81, COMMODORE VIC 20, and TANDY COLOUR COMPUTER in 32K with extended BASIC. Programs and text which are specifically for particular machines are indicated by the following symbols:

SPECTRUM 16K, 48K, 128, and +
COMMODORE 64 and 128
ACORN ELECTRON, BBC B and B+
DRAGON 32 and 64
ZX81
VIC 20
TANDY TRS80 COLOUR COMPUTER

TUMBLING DICE

At last, a computer game that's designed for several players. So get your family and friends together and start throwing the dice in this game of luck and skill

- RULES OF THE GAME
- STRATEGY
- DICE UDGS
- THROWING THE DICE
- SCORING

So far, nearly all the games in *INPUT*, whether arcade games, adventure games or strategy games, have pitted the player against the computer. With all of these games the main problem was to make the rules and the setting complex enough to give the player an enjoyable game, or, in the case of strategy games, to turn the computer into an intelligent and worthwhile adversary. This game is different. Instead of one person playing alone, this game is designed for up to six people playing against each other. The computer does not take part in the game itself. Instead, it keeps track of each player's score, makes sure no one cheats, and displays the score card—leaving the players to concentrate on the best strategy for winning the game.

The game is a computerized version of the popular dice game called Yacht. Yacht is an engrossing game combining luck and judgement as each player aims to make the highest score. The rules are quite simple. Each player throws five dice at a time (or rather, in this version, the computer throws the dice and displays them on the screen). If you don't like what comes up, you are allowed to have two more goes at throwing the dice and can choose how many of the dice to throw each time in an attempt to build up the best 'hand' you can. After the three goes you must enter the throw on the score card and the turn passes to the next player.

The options on the score card are:

Dice	**Score**
Ones	Total value of ones only
Twos	Total value of twos only
⋮	⋮
Sixes	Total value of sixes only
4 of a kind	Total of the four dice
Full house	Total of all five dice
Short run	15
Long run	30
Choice	Total of all five dice
Yacht	50

A short run is a run of four dice, say 2, 3, 4, 5, and a long run is a run of all five dice, either 1, 2, 3, 4, 5 or 2, 3, 4, 5, 6. A full house consists off three numbers of one kind plus a pair of any other number. Choice is a mixture of any dice, and Yacht is five of a kind.

Players must select a different category on each turn. To select a category, move the arrow up or down and press the space bar when the arrow points to your choice. If, at the end of the three throws, the dice cannot be fitted into any of the vacant categories, you have to choose which category to 'waste'. It is obviously best to waste one of the low scoring categories such as the ones or twos. However, towards the end of a round, you may be forced to waste some of the higher-scoring categories. In fact, it is good strategy to aim for the higher-scoring categories first, as these are more difficult to get.

The program is divided into three main sections, the initialization routine, the main game loop, and the subroutines or procedures called by the main loop.



INITIALIZATION

This section sets up the UDGs which display the dice, initializes the variables and asks for the names of the players.

```
20 LET Q$="..........": LET Z$="□□□
   □□□□□□□□": DIM C(13): FOR
   N=1 TO 13: READ C(N): NEXT N: DIM
   T(5): DIM R(5): DIM D(5)
30 FOR N=USR "A" TO USR "G"+7:
   READ A: POKE N,A: NEXT N
40 DATA 2,3,4,5,6,7,11,14,17,20,23,25,27
50 DATA 0,0,0,24,24,0,0,0
60 DATA 0,6,6,0,0,96,96,0
70 DATA 3,3,0,24,24,0,192,192
80 DATA 0,102,102,0,0,102,102,0
```



```
90 DATA 195,195,Ø,24,24,Ø,195,195
100 DATA 102,102,Ø,102,102,Ø,102,102
110 DATA Ø,24,48,96,255,96,48,24
120 PRINT AT 10,13;"YACHT": INK 1: PRINT
    AT 12,7;"HOW MANY PLAYERS""TAB
    11;"(1 TO 6)"
130 INPUT NP: LET NP=INT (NP): IF NP<1
    OR NP>6 THEN GOTO 130
140 DIM O(NP,12): DIM P(NP,12): DIM
    S(NP,5): DIM N$(NP,6): DIM Q(NP)
150 FOR N=1 TO NP: CLS : PRINT AT
    8,5;"PLAYER□"; (N);".""TAB 5;"WHAT'S
    YOUR NAME ?": INPUT W$: IF LEN
    W$>6 THEN LET W$=W$( TO 6)
160 LET N$(N)=Z$( TO 3-(LEN
    W$)/2)+W$: NEXT N
```

```
10 POKE53280,5:POKE53281,13:PRINTCHR$
   (147);CHR$(144);
20 DIMT(5),TR(5),D(5),A$(13),DC$(5)
30 FORZ=1TO6:READA:Z$=Z$+CHR$(A):
   NEXT
40 FORZ=1TO5:FORX=1TO5:READQ:
   DC$(Z)=DC$(Z)+CHR$(Q):NEXTX,Z:
   FORT=1TO12
50 READA$:A$(T)=A$+LEFT$("............",
   12-LEN(A$))+":":NEXTT
60 PRINTTAB(127);"HOW MANY PLAYERS
   (1-6) ?";
70 GETA$:IFA$<"1"ORA$>"6"THEN70
80 NP=VAL(A$):PRINTNP:DIMN$(NP),SC
   (NP),O(NP,13),P(NP,13),S(NP,5)
90 PRINTCHR$(149):FORN=1TONP:PRINT
   TAB(1);"PLAYER";N,:INPUT"NAME□";
   N$(N):NEXT
1150 DATA 17,157,157,157,157,157,32,32,32,
   32,32,209,32,32,32,32
1160 DATA 32,32,209,32,32,32,32,32,32,209,
   209,32,32,32,209
1170 DATA ONES,TWOS,THREES,FOURS,
   FIVES,SIXES,4 OF A KIND
1180 DATA FULL HOUSE,SHORT RUN,LONG
   RUN,CHOICE,YACHT
```

```
20 MODE2:VDU23,1;Ø;Ø;Ø;Ø:*FX11,Ø
30 DIMT(5),TR(5),D(5),A$(13)
40 FORT=1TO12:READA$:A$(T)=A$+
   STRING$((12-LEN(A$)),".")+":":NEXT
50 VDU23,224,Ø,Ø,Ø,24,24,Ø,Ø,Ø
60 VDU23,225,Ø,6,6,Ø,Ø,96,96,Ø
70 VDU23,226,3,3,Ø,24,24,Ø,192,192
80 VDU23,227,Ø,102,102,Ø,Ø,102,102,Ø
90 VDU23,228,195,195,Ø,24,24,Ø,195,195
100 VDU23,229,102,102,Ø,102,102,Ø,102,102
110 VDU23,230,Ø,24,48,96,255,96,48,24
120 COLOUR130:CLS:COLOUR7
130 PRINTTAB(2,14)"HOW MANY
    PLAYERS""□□□□□( 1 TO 6 )":
    REPEAT:NP=GET-48:UNTILNP>Ø AND
    NP<7
```



```
140 DIMN$(NP),SC(NP),O(NP,13),P(NP,13),S
    (NP,1Ø)
150 COLOUR128:CLS:COLOUR2:FORN=1TO
    NP
160 PRINTTAB(5,1Ø)"PLAYER□";N;"."
    "□WHAT'S YOUR NAME ?":INPUTTAB
    (7,14)N$(N):CLS:NEXT
```

```
10 CLS:X$=CHR$(13):DIMD$(6,4)
20 FORK=1TO6:FORJ=1TO3:FORL=1TO3:
   READA:D$(K,J)=D$(K,J)+CHR$(128+
   65*A):NEXT
30 D$(K,J)=D$(K,J)+CHR$(133):NEXT
40 D$(K,J)=STRING$(3,131)+CHR$(135):
   NEXT
50 DATA Ø,Ø,Ø,Ø,1,Ø,Ø,Ø,Ø,1,Ø,Ø,Ø,Ø,Ø,Ø,Ø,1
60 DATA 1,Ø,Ø,Ø,1,Ø,Ø,Ø,1,1,Ø,1,Ø,Ø,Ø,1,Ø,1
70 DATA 1,Ø,1,Ø,1,Ø,1,Ø,1,1,Ø,1,1,Ø,1,1,Ø,1
80 PRINT:PRINT" HOW MANY PLAYERS
   (1-6) ?";
90 A$=INKEY$:IF A$<"1" OR A$>"6"
   THEN 9Ø
100 PRINTA$:NP=VAL(A$):CLS
110 FORN=1TONP:PRINT@65,"PLAYER";N:
    PRINT"□WHAT'S YOUR NAME ?":INPUT
    N$(N)
120 CLS:NEXT
130 DIMO(NP,12),P(NP,12),S(NP,1Ø)
```

THE MAIN GAME

The structure of the game is very simple and consists of only these few lines:

```
170 FOR R=1 TO 5: FOR I=1 TO 12: FOR
    N=1 TO NP
180 BORDER 4: INK Ø: PAPER 4: CLS : PRINT
    AT 3,13;N$(N)
190 FOR M=5 TO 27: PRINT PAPER Ø;AT
    5,M;"□";AT 19,M;"□": NEXT M
200 FOR M=6 TO 18: PRINT PAPER Ø;AT
    M,5;"□";AT M,27;"□": NEXT M
210 GOSUB 240: PAUSE Ø: GOSUB 430
230 NEXT N: NEXT I: GOSUB 1290: NEXT R:
    STOP
```

```
100 FORR=1TO5:FORI=1TO12:FORN=1TO
    NP
110 PRINTCHR$(147);CHR$(31):GOSUB114Ø
120 PRINT
130 GOSUB17Ø:FORE=1TO15ØØ:NEXT
140 POKE5328Ø,6:POKE53281,14:GOSUB31Ø
150 POKE5328Ø,5:POKE53281,13:NEXTN,I:
    POKE5328Ø,2:POKE53281,1Ø:GOSUB94Ø
160 NEXTR:POKE5328Ø,Ø:POKE53281,11:
    PRINTCHR$(147);CHR$(5);"BYE NOW!":
    END
```



```
1140 PRINTSPC((4Ø–LEN(N$(N)))/2);
   N$(N):RETURN
```

```
170 FORR=1TO5:FORI=1TO12:FORN=1TO
   NP
180 COLOUR128:CLS:COLOUR6:PROCNAME
   (3)
190 VDU28,Ø,22,19,5:COLOUR132:CLS:VDU
   28,1,21,18,6:COLOUR13Ø:CLS:VDU26
200 PROCTHROW
210 FORE=1TO15ØØ:NEXTE
220 MODE1:VDU23,1;Ø;Ø;Ø;Ø:PROCSCORE:
   MODE2:NEXTN,I:MODE1:VDU23,1;Ø;Ø;Ø;Ø
230 PROCTABLE:MODE2:VDU23,1;Ø;Ø;Ø;Ø:
   NEXT:END
920 DEF PROCNAME(Y):PRINTTAB(1Ø–
   ((LEN(N$(N)))/2),Y)N$(N):ENDPROC
```

```
140 FORR=1TO5:FORI=1TO12:FORN=1TO
   NP
150 CLS:W=6:Y=2:GOSUB98Ø:GOSUB190
160 SOUND5Ø,3:FORE=1TO8ØØ:NEXT
170 CLS:GOSUB35Ø:CLS:NEXTN,I
180 CLS:GOSUB99Ø:NEXTR:END
980 PRINT@Y*32+W–((LEN(N$(N)))/2),
   N$(N):RETURN
```

There are three nested loops controlling the game. R is the number of rounds, I is the number of goes per round and N is the number of players. The routines called inside these loops throw the dice, print the score sheet and print the final score table. The Acorn, Dragon and Tandy also call a short routine to centre the name on the screen. The main routines are broken down yet again into smaller routines as you'll see in a moment.

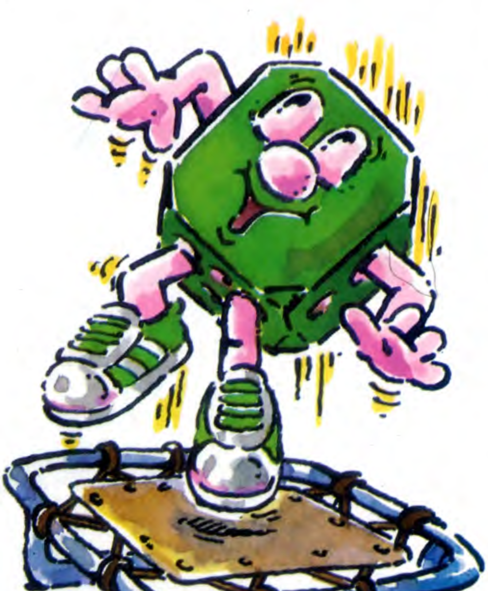
THROWING THE DICE

The first of the routines throws the dice and displays them on the screen, it calls two other routines which are given here as well. Add these to the last sections.

```
240 LET T=1: FOR D=1 TO 5: LET
   T(D)=INT (RND*6)+1: NEXT D
250 PRINT AT 6+T*3,7; "THROW□";T
260 GOSUB 118Ø
270 IF T=3 THEN GOTO 39Ø
280 LET C=1: FOR D=1 TO 5
290 PRINT AT 7+T*3,16+D*2;"?"
300 FOR J=1 TO 5Ø: NEXT J
310 LET A$=INKEY$: IF A$="" THEN
   GOTO 31Ø
320 IF A$="N" THEN BEEP .1,1Ø: GOTO
   36Ø
330 IF A$<>"Y" THEN GOTO 31Ø
340 BEEP .1,3Ø
350 LET R(C)=T(D): LET C=C+1
360 PRINT AT 7+T*3,16+D*2;"□": NEXT
   D
370 IF C=6 THEN GOSUB 42Ø: LET T=4:
   GOTO 4ØØ
380 FOR D=C TO 5: LET R(D)=INT
   (RND*6)+1: NEXT D: GOSUB 42Ø
390 LET T=T+1
400 IF T<>4 THEN GOTO 25Ø
410 RETURN
420 FOR D=1 TO 5: LET T(D)=R(D): NEXT
   D: RETURN
1180 FOR D=1 TO 5: PRINT PAPER 2;
   INK 6; BRIGHT 1;AT 6+T*3,16+
   D*2;CHR$ (143+T(D)): PAUSE 2: BEEP
   .Ø1, RND*4Ø: NEXT D: RETURN
```

```
170 T=1:FORD=1TO5:T(D)=INT
   (RND(1)*6)+1:NEXTD
180 PRINTTAB(1);"THROW:";SPC(2);T;:
   GOSUB1Ø4Ø
190 IFT=3THENT=4:PRINTTAB(12Ø):GOTO
   28Ø
200 C=1:PRINTTAB(165);:FORD=1TO5:
   PRINTSPC(5);"?";CHR$(157);
210 GETA$:IFA$<>"Y"ANDA$<>"N"
   THEN21Ø
220 POKE5328Ø,5:IFA$="N"THEN24Ø
230 TR(C)=T(D):C=C+1
240 PRINTCHR$(32);:NEXTD:PRINTCHR$
   (145)
250 IFC=6THENGOSUB3ØØ:T=4:GOTO28Ø
260 FORD=CTO5:TR(D)=INT(RND
   (1)*5)+1:NEXTD
270 GOSUB3ØØ:T=T+1
280 IFT<4THEN18Ø
290 RETURN
300 FORD=1TO5:T(D)=TR(D):NEXT:
   RETURN
1040 PRINTCHR$(17);CHR$(17);CHR$(29);:
   FORX=1TO5:D=T(X)
1050 PRINTCHR$(18);:FORQ=1TO5:PRINT
   CHR$(157);:NEXTQ
1060 IFD=1THENPRINTDC$(1);Z$;DC$(1);
   Z$;DC$(3);Z$;DC$(1);Z$;DC$(1);
1070 IFD=2THENPRINTDC$(2);Z$;DC$(1);
   Z$;DC$(1);Z$;DC$(1);Z$;DC$(4);
1080 IFD=3THENPRINTDC$(4);Z$;DC$(1);
   Z$;DC$(3);Z$;DC$(1);Z$;DC$(2);
1090 IFD=4THENPRINTDC$(5);Z$;DC$(1);
   Z$;DC$(1);Z$;DC$(1);Z$;DC$(5);
1100 IFD=5THENPRINTDC$(5);Z$;DC$(1);
   Z$;DC$(3);Z$;DC$(1);Z$;DC$(5);
1110 IFD=6THENPRINTDC$(5);Z$;DC$(1);
   Z$;DC$(5);Z$;DC$(1);Z$;DC$(5);
1120 PRINTCHR$(145);CHR$(145);CHR$
   (145);CHR$(145);
1130 FORZ=1TO6:PRINTCHR$(29);:NEXTZ,X:
   PRINT:RETURN
```

```
240 DEF  PROCTHROW
250 VDU 23, 1; Ø; Ø; Ø; Ø: T=1
260 FOR D=1 TO 5: T(D)=RND (6): NEXT
270 REPEAT
280 COLOUR Ø: COLOUR 13Ø: PRINT TAB (1,
   8+T*3) "THROW:";T
290 PROCDICE: IF T=3 THEN T=4: GOTO
   38Ø ELSE C=1
300 FOR D=1 TO 5: COLOUR 13Ø: COLOUR
   8: PRINT TAB (8+D*2, 9+T*3) "?"
310 A$=GET$: IF A$= "N" THEN SOUND
   1, –15, 2, 2: GOTO 34Ø
320 IF A$<> "Y" THEN 31Ø ELSE SOUND
   1, –15, 15Ø, 4
330 TR(C)=T(D): C=C+1
```





```
340 PRINTTAB (8+D*2, 9+T*3) "□": NEXT
350 IF C=6 THEN PROCTRANS: T=4: GOTO
    380
360 FOR D=C TO 5: TR(D)=RND (6): NEXT
370 PROCTRANS: T=T+1
380 UNTIL  T=4: ENDPROC
390 DEF PROCTRANS
400 FORD=1 TO 5: T(D)=TR(D): NEXT:
    ENDPROC
910 DEF PROCDICE: COLOUR 129: COLOUR
    3: FOR D=1 TO 5: PRINT TAB (8+D*2,
    8+T*3) CHR$(223+T(D)): FOR E=1 TO
    8: NEXT: SOUND 0, -15, 0, 1: NEXT:
    ENDPROC
```

```
190 T=1:FORD=1TO5:T(D)=RND(6):NEXT
200 PRINT@64*T+64,"THROW:";T;
210 GOSUB970:IFT=3 THEN310
220 C=1:FORD=1TO5
230 PRINT@288,TAB(9+D*4)"?"
240 A$=INKEY$:IFA$<>"N" AND
    A$<>"Y" THEN240
250 IF A$="N" THENSOUND10,1:GOTO270
260 SOUND100,1:TR(C)=T(D):C=C+1
270 NEXTD:PRINT@288
280 IFC=6GOSUB340:RETURN
290 FORD=C□TO5:TR(D)=RND(6):NEXTD
300 GOSUB340
310 T=T+1
320 IF T<>4 THEN 200
330 RETURN
340 FORD=1TO5:T(D)=TR(D):NEXTD:RETURN
970 FORD=1TO5:FORG=1TO4:PRINT@
    136+G*32+D*4,D$(T(D),G);:NEXTG,D:
    RETURN
```

Five dice are thrown at first and these are displayed on the screen using the routine at Line 1180 on the Spectrum, Lines 1040 to 1130 on the Commodore, 910 on the Acorn and 970 on the Dragon and Tandy. You are then given the chance to select the dice by pressing Y for the ones you wish to keep and N for the ones you want to throw again.

The initial numbers of the five dice are stored in array T(). After the first throw, the ones you wish to keep are put into a temporary array R() and this is made up to five again with random numbers. The R() array is then transferred back into T() using the one-line routine at 420 on the Spectrum, 300 on the Commodore, 400 on the Acorn and 340 on the Dragon and Tandy. These are displayed on the screen once more and the process is repeated for your next throw.

THE SCORE CARD

The vast majority of the program is concerned with calculating the score and checking the entries on the score card.

```
430 BORDER 0: PAPER 0: INK 6: CLS
440 PLOT 4,4: DRAW 0,167: DRAW 124,0:
    DRAW 0,-167: DRAW -124,0
450 PRINT INK 5;AT 1,5;N$(N); INK 4; AT
    2,1;"**SCORE SHEET**"
460 RESTORE 1280: FOR M=4 TO 17: READ
    A$: PRINT AT M,1;A$;Q$( TO 11-LEN
    A$);: IF M<>16 THEN PRINT ":"
470 NEXT M
480 GOSUB 530
490 GOSUB 560: GOSUB 530
500 PRINT FLASH 1;AT 20,18;"ANY KEY
    TO";AT 21,18;"CONTINUE□□"
510 LET A$=INKEY$: IF A$="" THEN
    GOTO 510
520 RETURN
530 FOR D=1 TO 12: IF P(N,D)=1 THEN
    PRINT AT 3+D,13;"X"
540 IF O(N,D)<>0 THEN PRINT AT
    3+D,13;O(N,D)
550 NEXT D: LET C=0: FOR D=1 TO 12:
    LET C=C+O(N,D): NEXT D: PRINT AT
    17,13;C: RETURN
560 PRINT AT 8,18;"ROUND□";R;AT
    9,18;"SECTION□";I
570 PRINT AT 2,18;"FINAL SET=□": LET
    T=-1: GOSUB 1180
580 PRINT AT 5,18;"SELECT SCORE";AT
    6,18;"GROUP."
590 LET A=4
600 PRINT AT A,15;CHR$ 150
610 LET B$=INKEY$: IF B$="" THEN
    GOTO 610
620 IF B$="□" THEN LET A=A-3:
    GOTO 710
630 IF B$="K" THEN GOTO 650
640 IF B$<>"M" THEN GOTO 610
650 PRINT AT A,15;"□"
660 IF B$="K" AND A=4 THEN GOTO 600
670 IF B$="M" AND A=15 THEN GOTO
    600
680 IF B$="M" THEN LET A=A+1
690 IF B$="K" THEN LET A=A-1
700 BEEP .01,5: GOTO 600
710 PRINT AT A+3,15;"□": IF
    P(N,A)<>0 THEN GOTO 1240
720 IF A>6 THEN GOTO 780
730 LET C=0
740 FOR D=1 TO 5: IF T(D)=A THEN LET
    C=C+1
750 NEXT D
760 LET O(N,A)=C*A
770 LET P(N,A)=1: RETURN
780 IF A=11 THEN FOR D=1 TO 5: LET
    O(N,11)=O(N,11)+T(D): NEXT D: LET
    P(N,11)=1: RETURN
790 FOR D=1 TO 5: LET D(D)=0: NEXT D:
    LET B=0: FOR E=1 TO 6: LET C=0:
    FOR D=1 TO 5: IF T(D)=E THEN LET
    C=C+1
800 NEXT D: IF C<>0 THEN LET B=B+1
810 NEXT E
820 LET G=1: FOR F=1 TO 6: GOSUB
    1250: IF C<>0 THEN LET D(G)=F: LET
    G=G+1
830 NEXT F
840 LET P(N,A)=1: IF A=7 THEN GOTO 950
850 IF A=8 THEN GOTO 1010
860 IF A=9 THEN GOTO 1050
870 IF A=10 THEN GOTO 1120
890 IF A=12 THEN GOTO 1160
950 IF B>2 THEN GOTO 1190
960 IF B=1 THEN GOSUB 1270: LET
    O(N,7)=C: RETURN
970 LET F=1
980 GOSUB 1250: LET F=F+1: IF C<>4
    AND F<>7 THEN GOTO 980
990 IF C<4 THEN GOTO 1190
1000 LET O(N,7)=4*(F-1): RETURN
1010 IF B<>2 THEN GOTO 1190
1020 LET F=D(1): GOSUB 1250: IF C=3
    THEN GOTO 1040
1030 LET F=D(2): GOSUB 1250: IF C<>3
    THEN GOTO 1190
1040 LET O(N,8)=0: FOR G=1 TO 5: LET
    O(N,8)=O(N,8)+T(G): NEXT G: RETURN
1050 IF B<>4 THEN GOTO 1080
1060 GOSUB 1270: IF C<>18 AND
    C<>10 AND C<>14 OR (C=14 AND
    D(4)=6) THEN GOTO 1190
1070 LET O(N,9)=15: RETURN
1080 IF B<>5 THEN GOTO 1190
1090 GOSUB 1270: IF C=15 OR C=16 OR
    C=19 THEN GOTO 1070
1100 IF C<>20 THEN GOTO 1190
1110 GOTO 1070
1120 IF B<>5 THEN GOTO 1190
1130 GOSUB 1270: IF C=15 OR C=20
    THEN GOTO 1150
1140 GOTO 1190
1150 LET O(N,10)=30: RETURN
1160 IF B<>1 THEN GOTO 1190
1170 LET O(N,12)=50: RETURN
1190 BEEP .5,5: PRINT AT 20,18;
    "ILLEGAL !!";AT 21,18;"WASTE ?"
1200 LET A$=INKEY$: IF A$="" THEN
    GOTO 1200
1210 IF A$="N" THEN PRINT AT 20,18;"□
    □□□□□□□□□□";AT 21,18;"□
    □□□□□□": LET P(N,A)=0: GOTO
    590
1220 IF A$<>"Y" THEN GOTO 1200
1230 LET P(N,A)=1: RETURN
1240 BEEP .5,5: PRINT AT 20,18;"SECTION
    FILLED": FOR H=1 TO 300: NEXT H:
    PRINT AT 20,18;"□□□□□□□□□□
    □□□□□": GOTO 590
1250 LET C=0: FOR D=1 TO 5: IF T(D)=F
    THEN LET C=C+1
1260 NEXT D: RETURN
```

```
1270 LET C=Ø: FOR D=1 TO B: LET
   C=C+D(D): NEXT D: RETURN
1280 DATA "ONES","TWOS","THREES",
   "FOURS","FIVES","SIXES","4 OF A
   KIND","FULL HOUSE","SHORT RUN",
   "LONG RUN","CHOICE","YACHT","□
   □□□□□□□□□□□","TOTAL"
```

```
310 PRINTCHR$(147);CHR$(5):GOSUB114Ø:
   PRINT
320 FORZ=1TO51:PRINTCHR$(18);
   CHR$(31);CHR$(32);:NEXTZ
330 PRINT"** SCORE SHEET **";
340 FORZ=1TO51:PRINTCHR$(18);CHR$
   (31);CHR$(32);:NEXTZ:PRINT:PRINT:
   PRINT
350 FORT=1TO12:PRINT,CHR$(149);A$(T):
   NEXTT
360 PRINTTAB(5Ø);CHR$(144);"TOTAL"
370 GOSUB42Ø:GOSUB49Ø:GOSUB42Ø
380 PRINTCHR$(19);:FORZ=1TO24:PRINT
   CHR$(17);:NEXTZ
390 PRINTCHR$(31);SPC(2);"PLEASE PRESS
   ANY KEY TO CONTINUE...";
400 GETA$:IFA$=""THEN4ØØ
410 RETURN
420 PRINTCHR$(19);CHR$(5);:FORZ=1TO9:
   PRINTCHR$(17);:NEXTZ:FORD=1TO12
430 IFO(N,D)<>ØTHENPRINTSPC(3Ø);
   O(N,D):GOTO46Ø
440 IFP(N,D)=1THENPRINTSPC(31);
   CHR$(214):GOTO46Ø
450 PRINT
460 NEXTD:C=Ø:FORD=1TO12
470 C=C+O(N,D):NEXTD:PRINT
480 PRINTSPC(3Ø);C:S(N,R)=C:RETURN
490 PRINTCHR$(19);CHR$(5);TAB(24Ø);
500 PRINT"ROUND";R;"TURN";I;
510 PRINT"FINAL SET =";:FORD=1TO5:
   PRINTT(D);CHR$(157);:NEXTD
520 PRINT,SPC(7);CHR$(144);"PLEASE
   SELECT SCORE GROUP:";
530 A=1:K=1353+(A*4Ø):POKEK,62:POKE
   K+54272,1
540 K=1353+(A*4Ø):POKEK,94-PEEK(K):
   POKEK+54272,1
550 GETA$:IFA$=CHR$(32)THEN61Ø
560 IFA$<>CHR$(145)ANDA$<>CHR$
   (17)THEN54Ø
570 POKEK,32:A=A+(A$=CHR$(145))
   -(A$=CHR$(17))
580 IFA=ØTHENA=12
590 IFA=13THENA=1
600 GOTO54Ø
610 POKEK,32
620 IFP(N,A)<>ØTHEN92Ø
630 IFA<>11THEN65Ø
640 FORD=1TO5:O(N,A)=O(N,A)+T(D):
   NEXT:P(N,A)=1:RETURN
650 IFA>6THEN69Ø
660 C=Ø:FORD=1TO5:IFT(D)=ATHEN
   C=C+1
670 NEXT:IFC=ØTHEN87Ø
680 O(N,A)=C*A:P(N,A)=1:RETURN
690 FORD=1TO5:D(D)=Ø:NEXT:B=Ø:FOR
   F=1TO6:GOSUB1Ø2Ø
700 IFC<>ØTHENB=B+1
710 NEXT:G=1:FORF=1TO6:GOSUB1Ø2Ø:
   IFC<>ØTHEND(G)=F:G=G+1
720 NEXT:P(N,A)=1:ONA-6GOTO73Ø,78Ø,
   82Ø,84Ø,915,86Ø
730 IFB>2THEN87Ø
740 F=1
750 GOSUB1Ø2Ø:F=F+1:IFC<4ANDF<7
   THEN75Ø
760 IFC<4THEN87Ø
770 O(N,A)=4*(F-1):RETURN
780 IFB<>2THEN87Ø
790 F=D(1):GOSUB1Ø2Ø:IFC=3THEN81Ø
800 F=D(2):GOSUB1Ø2Ø:IFC<>3THEN
   87Ø
810 GOTO64Ø
820 IFB>3THENB=4:GOSUB1Ø1Ø:IFC=1Ø
   ORC=14ORC=18THENO(N,A)=15:
   RETURN
830 GOTO87Ø
840 GOSUB1Ø1Ø:IFB=5AND(C=15OR
   C=2Ø)THENO(N,A)=3Ø:RETURN
850 GOTO87Ø
860 IFB=1THENO(N,A)=5Ø:FORE=1TO
   1ØØØ:NEXT:RETURN
870 PRINTCHR$(19);"NO SCORE-WASTE
   IT?"
880 GETA$:IFA$<>"Y"ANDA$<>"N"
   THEN88Ø
890 PRINTCHR$(19);:FORZ=1TO2Ø:PRINT
   CHR$(32);:NEXTZ
900IFA$="N"THENP(N,A)=Ø:GOTO53Ø
910 P(N,A)=1:RETURN
915 GOSUB1Ø1Ø:O(N,A)=C:RETURN
920 PRINTCHR$(19);"SECTION FILLED":FOR
   E=1TO15ØØ:NEXTE
930 PRINTCHR$(19);:FORZ=1TO14:PRINT
   CHR$(32);:NEXT:GOTO53Ø
1010 C=Ø:FORD=1TOB:C=C+D(D):NEXT
   D:RETURN
1020 C=Ø:FORD=1TO5:IFT(D)=FTHEN
   C=C+1
1030 NEXT:RETURN
```

```
410 DEF PROCSCORE
420 COLOUR128:COLOUR1:MOVE64Ø,8:GCOL
   Ø,2:DRAW64Ø,1Ø15:DRAW1271,1Ø15:
   DRAW1271,8:DRAW64Ø,8:VDU28,21,31,38,
   1
430 PROCNAME(Ø):COLOUR2:PRINT
   "****SCORE SHEET****"
440 COLOUR3:FOR T=1TO12:PRINTA$(T)':
   NEXT:PRINTA$(T)
450 PRINT:COLOUR2:PRINT"TOTAL.......:"
```



```
460 PROCTOTAL:PROCSCSEL:PROCTOTAL
470 COLOUR129:COLOURØ:PRINTTAB(Ø,1Ø)
   "ANY KEY TO CONTINUE"
480 A$=GET$:ENDPROC
490 DEF PROCTOTAL:VDU28,21,31,38,1:
   COLOUR1:FORD=1TO13
500 IFO(N,D)<>ØTHENPRINTTAB(13,
   1+D*2);O(N,D)ELSEIFP(N,D)=1THEN
   PRINTTAB(13,1+D*2)"X"
510 NEXTD:C=Ø:FORD=1TO12:C=C+O
   (N,D):NEXTD:PRINTTAB(13,29);C:VDU26:
   ENDPROC
520 DEF PROCSCSEL
530 COLOUR135:COLOURØ:PRINTTAB(Ø,3)
   "ROUND□";R;"□SECTION□";I:
   COLOUR128
540 COLOUR2:PRINTTAB(Ø,5)"FINAL SET="
550 T=-1:PROCDICE:COLOUR7:PRINT:
   PRINT"SELECT SCORE GROUP"
560 COLOUR3:COLOUR128:A=1
570 VDU31,37,2+A*2,23Ø
580 B=GET:IFB=32THEN66Ø
590 IF B<>58 AND B<>47 THEN 58Ø
600 PRINTTAB(37,2+A*2)"□"
610 IFB=47THENA=A+1 ELSE A=A-1
620 IFA=ØTHENA=12
630 IFA=13THENA=1
640 SOUND1,-15,22Ø,1
650 GOTO57Ø
660 PRINTTAB(37,2+A*2)"□":IFP(N,A)
   <>ØTHEN89Ø
670 IFA<>11THEN69Ø
680 FORD=1TO5:O(N,A)=O(N,A)+T(D):
   NEXT:P(N,A)=1:ENDPROC
690 IFA>6THEN72Ø
700 C=Ø:FORD=1TO5:IFT(D)=A□THEN
   C=C+1
710 NEXT:IF C=Ø THEN 84Ø ELSE O(N,A)=
   C*A:P(N,A)=1:ENDPROC
```



```
720 FORD=1TO5:D(D)=0:NEXT:B=0:FOR
    F=1TO6:PROCMAJ:IFC<>0 THEN B=B+1
730 NEXT:G=1:FORF=1TO6:PROCMAJ:IF
    C<>0THEND(G)=F:G=G+1
740 NEXT:P(N,A)=1:ON A-6 GOTO 750,
    780,800,820,10,830
750 IFB>2THEN840
760 F=1:REPEAT:PROCMAJ:F=F+1:UNTIL
    C>=4 OR F=7:IF C<4THEN840
770 O(N,A)=4*(F-1):ENDPROC
780 IFB<>2THEN 840 ELSE F=D(1):
    PROCMAJ:IFC=3THEN790 ELSE F=D(2):
    PROCMAJ:IFC<>3THEN840
790 GOTO680
800 IF B<>4 THEN 812 ELSE PROCTOTE:IF
    C=10 OR (C=14 AND D(4)<>6) OR
    C=18 THEN O(N,A)=15:ENDPROC
810 GOTO840
812 IF B<>5 THEN 840 ELSE PROCTOTE:IF
    C=20 OR C=15 OR C=16 OR C=19
    THEN O(N,A)=15:ENDPROC
820 PROCTOTE:IF B=5 AND(C=15 OR
    C=20) THEN O(N,A)=30:ENDPROC ELSE 840
830 IF B=1 THEN O(N,A)=50:T=INKEY
    (200):ENDPROC
840 SOUND1,-15,20,10:PRINTTAB(0,10)
    "ILLEGAL! WASTE IT?"
850 A$=GET$
860 IFA$="N"THENPRINTTAB(0,10)
    STRING$(18,"□");:P(N,A)=0:GOTO560
870 IFA$<>"Y"THEN850
880 P(N,A)=1:ENDPROC
890 SOUND1,-15,5,10:PRINTTAB(2,10)
    "SECTION FILLED";:FORE=1TO1500:
    NEXTE:PRINTTAB(2,10)"□□□□□□□
    □□□□□□□□□□□";:GOTO560
900 ENDPROC
1070 DEF PROCMAJ:C=0:FORD=1TO5:IFT
    (D)=F□THEN C=C+1
1080 NEXT:ENDPROC
1090 DEF PROCTOTE:C=0:FORD=1TO5:
    C=C+D(D):NEXTD:ENDPROC
1100 DATA ONES,TWOS,THREES,FOURS,
    FIVES,SIXES,4 OF A KIND,FULL HOUSE,
    SHORT RUN,LONG RUN,CHOICE,YACHT
```

```
350 CLS
360 W=6:Y=0:GOSUB980:PRINT
    "****SCORE SHEET****"
370 PRINT"ONES........:"X$"TWOS
    ........:"X$"THREES......:"X$"FOURS
    .......:"X$"FIVES.......:"X$"SIXES
    .......:"
380 PRINT"4 OF A KIND.:"X$"FULL HOUSE
    ..:"X$"SHORT RUN...:"X$"LONG RUN
    ....:"X$"CHOICE......:"X$"YACHT
    .......:"X$
390 PRINT"TOTAL.......:";
400 GOSUB460
410 GOSUB490:GOSUB460
420 PRINT@467,"any key to";:PRINT@500,
    "continue";
430 SOUND60,1
440 A$=INKEY$:IFA$="" THEN440
450 RETURN
460 FORD=1TO12:IFP(N,D)=1 THEN
    PRINT@45+D*32,"X";
470 IFO(N,D)<>0 THENPRINT@45+D*32,
    O(N,D);
480 NEXTD:C=0:FORD=1TO12:C=C+O
    (N,D):NEXT:PRINT@493,C;:RETURN
490 PRINT@53,"ROUND"R;:PRINT@84,
    "SECTION"I;
500 PRINT@115,"FINAL SET=";
510 FORD=1TO5:FORG=1TO4:
    PRINT@111+G*32-(D>3)*118+4*D,
    D$(T(D),G);:NEXTG,D
520 PRINT@403,"SELECT GROUP";
530 A=1
540 PRINT@49+32*A,CHR$(95);
550 B$=INKEY$:IF B$<>"□" AND
    B$<>"↑" AND B$<>CHR$(10)
    THEN550
560 IFB$="□" THEN620
570 PRINT@49+32*A,"□";
580 IFB$="↑" AND A>1 THENA=A-1
590 IFB$=CHR$(10) AND A<12 THEN
    A=A+1
600 SOUND200,1
610 GOTO540
620 PRINT@49+32*A,"□";:IF
    P(N,A)<>0 THEN950
630 IFA>6 THEN700
640 C=0
650 FORD=1TO5:IFT(D)=A□THEN
    C=C+1
660 NEXTD
670 O(N,A)=C*A
680 P(N,A)=1
690 RETURN
700 IFA=11 THENFORD=1TO5:O(N,11)=O
    (N,11)+T(D):NEXT:P(N,11)=1:RETURN
710 FORD=1TO5:D(D)=0:NEXT:B=0:FOR
    E=1TO6:C=0:FORD=1TO5:IFT(D)=E
    THENC=C+1
720 NEXTD:IFC<>0 THENB=B+1
730 NEXTE
740 G=1:FORF=1TO6:GOSUB1140:IF
    C<>0THEND(G)=F:G=G+1
750 NEXTF
760 P(N,A)=1:A=A-6:ONA GOTO770,810,
    830,870,960,890
770 IFB>2 THEN900 ELSEIFB=1 GOSUB
    1160:O(N,7)=C*4:RETURN
780 F=1
790 GOSUB1140:F=F+1:IF C<>4 AND
    F<>7 THEN790 ELSEIFC<4 THEN 900
800 O(N,7)=4*(F-1):RETURN
810 IFB<>2 THEN900 ELSEF=D(1):
    GOSUB1140:IFC=3THEN820 ELSE
    F=D(2):GOSUB1140:IFC<>3THEN900
820 FORD=1TO5:O(N,8)=O(N,8)+T(D):
    NEXT:RETURN
830 IFB<>4 THEN850 ELSEGOSUB1160:IF
    C<>18ANDC<>10ANDC<>14OR
    (C=14ANDD(4)=6) THEN900
840 O(N,9)=15:RETURN
850 IFB<>5THEN900 ELSEGOSUB1160:IF
    C<>20ANDC<>15ANDC<>16AND
    C<>19 THEN900
860 GOTO840
870 IFB<>5 THEN900 ELSEGOSUB1160:IF
    C<>20ANDC<>15 THEN900
880 O(N,10)=30:RETURN
890 IFB<>1 THEN900 ELSEO(N,12)=50:
    SOUND5,8:FORE=1TO700:NEXTE:
    RETURN
900 SOUND20,1:PRINT@432,"illegal. WASTE?";
910 A$=INKEY$:IF A$<>"Y" AND
    A$<>"N" THEN910
920 PRINT@432,"□□□□□□□□□□
    □□□□□□□";
930 IFA$="N"THENP(N,A+6)=0:GOTO530
940 P(N,A+6)=1:RETURN
950 SOUND5,1:PRINT@433,"section filled";:
    FORE=1TO700:NEXT:PRINT@433,"□□
    □□□□□□□□□□□□□□";:GOTO530
960 RETURN
1140 C=0:FORD=1TO5:IFT(D)=F□THEN
    C=C+1
1150 NEXTD:RETURN
1160 C=0:FORD=1TOB:C=C+D(D):NEXT
    D:RETURN
```

The first part of this section up to Line 520 on the Spectrum, 410 on the Commodore, 480 on the Acorn and 450 on the Dragon and Tandy, prints out the blank score card and calls two other routines to fill in the card and accept the latest entry. The following routine,

which takes up the next three lines, is the one which fills in the card with your previous score, and adds up the current total. Following this is a large section which accepts and validates your choice.

The first few lines display details of the round and section, and re-display your dice. The next part detects which keys are pressed and moves the cursor up and down the score card accordingly. The Spectrum version uses the M and K keys, to move the cursor, the Acorn uses : and / and the Commodore, Dragon and Tandy use the up and down arrow keys.

As soon as you press the space bar the choice is accepted and the position of the cursor is stored in the variable A. This variable conveniently points to your chosen category; A=1 for Ones, down to A=11 for Yacht. The value of A is used throughout the remainder of the routine to work out your score. Another important variable is B which holds the number of *different* dice in your throw—so B=2 for a full house and B=5 for a long run, for example.

So, depending on the value of A, different routines are called to check that your dice really do correspond to the category you've chosen. Assuming they do, the score is worked out and entered onto the score card. If they do not match, then the computer prints ILLEGAL! and you are asked if you want to waste this category. If you answer Y, an X will be printed instead of a score, and if you answer N you can go on to choose another category. The computer will also check if the category is already filled.



THE FINAL SCORE

The last section simply prints up the total score for each player at the end of each round. The numbers of the rounds are printed along the top and the names of the players down the side. There's also a total score for each player.

```
1290 BORDER 7: PAPER 7: INK 3: CLS
1300 PRINT "**********SCORE TABLE*********
   **"
1310 PRINT AT 7,Ø: FOR D=1 TO NP: PRINT
   N$(D);"□:"'': NEXT D
1320 PRINT INK 1;AT 3,Ø;"PLAYER"; INK
   2;AT 3,13;"R□O□U□N□D"
1330 PRINT AT 5,12;: FOR D=1 TO 5: PRINT
   INK 2;TAB (4+D*4);D;: NEXT D: PRINT
   INK Ø;"□□□TOTE"
1340 FOR D=1 TO NP
1360 LET C=Ø: FOR E=1 TO 12: LET
   C=C+O(D,E): NEXT E: LET S(D,R)=C:
   NEXT D
1370 FOR D=1 TO NP: FOR E=1 TO R
1380 PRINT INK 4;AT 6+D*2,4+E*4;
   S(D,E): NEXT E: NEXT D
1390 FOR D=1 TO NP: LET C=Ø: FOR E=1
   TO R: LET C=C+S(D,E): NEXT E: PRINT
   AT 6+D*2,28;C: NEXT D
1400 PRINT #1; INVERSE 1; INK
   Ø;"□□□□□□□ANY KEY TO
   CONTINUE□□□□□□□"
1410 LET A$=INKEY$: IF A$="" THEN
   GOTO 1410
1420 FOR E=1 TO NP: FOR D=1 TO 12:
   LET O(E,D)=Ø: LET P(E,D)=Ø: NEXT D:
   NEXT E
1430 RETURN
```

```
940 PRINTCHR$(147);CHR$(144);"**********
   ***SCORE TABLE**************"
950 FORD=1TONP:PRINTN$(D),:C=Ø
960 FORZ=1TO5:PRINTTAB(5+Z*5);S(D,Z);:
   C=C+S(D,Z):NEXT
970 PRINTCHR$(5);C:NEXT
980 PRINTTAB(41);"PLEASE PRESS ANY KEY
   FOR NEXT ROUND..."
990 GETA$:IFA$=""THEN990
1000 FORE=1TONP:FORD=1TO12:
   O(E,D)=Ø:P(E,D)=Ø:NEXTD,E:RETURN
```

```
930 DEF PROCTABLE
940 COLOUR3:PRINT''"***************SCORE
   TABLE**************":VDU28,Ø,28,39,4
950 COLOUR129:CLS:COLOUR2
960 PRINTTAB(Ø,7);:FORD=1TONP:PRINT
   N$(D):PRINT:NEXT
970 PRINTTAB(16,3)"R□O□U□N□D"
980 FORD=1TONP
990 C=Ø:FORE=1TO12:C=C+O(D,E):NEXT:
   S(D,R)=C : NEXT
1000 FORD=1TO5:PRINTTAB(8+D*4,5);D;:
   NEXTD
1010 COLOUR3:PRINT"□□TOTAL": COLOURØ
1020 FORD=1TONP:FORE=1TOR:PRINTTAB
   (6+E*4,5+D*2);S(D,E):NEXTE,D
1030 FORD=1TONP:C=Ø:FORE=1TOR:
   C=C+S(D,E):NEXTE:PRINTTAB(31,
   5+D*2);C:NEXTD
1040 COLOUR13Ø:VDU26:PRINTTAB(8,3Ø)
   "Any key for next round.":*FX21,Ø
1050 A$=GET$
1060 FORE=1TONP:FORD=1TO12:
   O(E,D)=Ø:P(E,D)=Ø:NEXTD,E:ENDPROC
```

```
990 CLS:PRINT"**********SCORE TABLE*******
   ***"
1000 PRINT@128:FORD=1TONP:PRINTN$
   (D):NEXT
1010 PRINT@75,"R□O□U□N□D"
1020 FORD=1TONP
1030 C=Ø:FORE=1TO12:C=C+O(D,E):
   NEXT:S(D,R)=C:NEXTD
1040 FORD=1TO5:PRINT@98+D*4,D;:
   NEXT
1050 PRINT"□TOTAL"
1060 FORD=1TONP
1070 FORE=1TOR
1080 PRINT@129+D*32+E*4,S(D,E);:NEXT
   E,D
1090 FORD=1TONP:C=Ø:FORE=1TOR:
   C=C+S(D,E):NEXTE:PRINT@153+
   D*32,C;:NEXTD
1100 PRINT@448,"ANY KEY FOR NEXT
   ROUND.";
1110 A$=INKEY$:IFA$="" THEN1110
1120 FORE=1TONP:FORD=1TO12:
   O(E,D)=Ø:P(E,D)=Ø:NEXTD,E
1130 RETURN
```



COMMODORE HI-RES GRAPHICS-3

Here is a total revision of the Commodore C64 Hi-Res program, superceding the previous articles and supplying the data in modular and machine-code form

In the first two parts of this article (see pages 748 to 751 and 872 to 877) you saw how to add graphics commands to supplement your Commodore 64's BASIC. Now you can add the rest of the commands to give your computer a complete set of graphics commands which will handle all of the Commodore 64's graphics programs given in *INPUT*.

Unfortunately it was discovered that due to the extremely advanced nature of the Hi-Res program as a whole too many bugs had crept into it to allow the normal practice of publishing errata corrections. So this article includes a complete revision of the Hi-Res program from the beginning.

SYNTAX

With all computer commands it is important to get the syntax exactly right. And *INPUT*'s hi-res graphics instructions are no exception. Each command must be followed by the right number of parameters. These must have values in the right range and be in the right order, otherwise strange things may start to happen—or you may just get an error message.

And don't forget to prefix all Commodore 64 graphics commands published in *INPUT* with an @. This helps the computer to identify the new commands that you've added to BASIC more quickly. The words @LOWCOL and @HICOL also have to be closed up with no space in the middle.

The instruction @HIRES takes two parameters. The first specifies the plotting colour to be used and the second specifies the background colour. They are specified by the logical colour numbers given in the Commodore Users' Manual.

The @COLOUR should also be followed by two parameters. The first specifies the border colour to be used and the second specifies the background colour of the low-resolution screen. Again the colour numbers in the Commodore Users' Manual are used.

The syntax used by @NRM, @CSET, @MULTI, @LOWCOL, @HICOL, @PLOT, @LINE, @REC and @BLOCK are given on pages 872 to 877.

The @CIRCLE command allows you to draw circles and ellipses of varying sizes. It takes five parameters. The first two specify the X and Y coordinates of centre of the circle or ellipse respectively.

The next pair specify the X and Y radii of the shape. By varying these you can produce different-shaped ellipses. But due to the rectangular shape of the screen, equal numbers here will not produce a circle. Indeed, different ratios are required to produce the same shaped ellipses on the @MULTI colour and @HIRES screens. So if you want to draw a circle you will have to experiment with the parameters a bit.

The last parameter specifies the plot type. For details see the @PLOT command on page 874.

It is also possible to draw part of a circle or an ellipse using the @ARC command. This takes eight parameters.

The first two are the X and Y coordinates of the shape which @ARC is drawing a section of. The next pair define the beginning and end angles of the arc. The fifth parameter specifies the interval between the dots used to make up the arc—a 1 here gives a solid line.

This works slightly differently from Simons' BASIC which joins up the different points if the line is not solid. So you will have to modify the program on page 369 and draw in the cat's ears with a @LINE command.

The sixth and seventh parameters specify the X and Y radii. And the last parameter gives the plot type.

The opposite of @ARC is @ANGL. This draws in the radii, but omits the circumference of the shape. It takes six parameters.

Again, the first two are X and Y parameters of the centre of the shape. The third parameter is the angle of the radius required measured clockwise, in degrees from the vertical position.

The next two are the horizontal and vertical radii again. And the last one is, as always, the plot type.

The @PAINT command fills an area of the screen with colour. The area to be filled must be completely enclosed by a line, otherwise the whole screen will be filled.

@PAINT takes three parameters. The first two are the X and Y coordinates of any point within the area to be filled—but don't specify a point on the edge or you might run into some problems. The @FLASH command flashes a specified colour on the screen from normal to reverse field and back again. It takes two parameters.

The first specifies the colour to be flashed. The second specifies the speed at which the flash is to take place. Speeds are defined by any number between 1 and 255.

Naturally, the @OFF command which switches the flash off needs no parameters. The @TEST command looks at a pixel at a specified location and returns the type of dot plotted there in memory location 2. So, to use this command in a BASIC program you must follow it by a PEEK (2). If no dot has been plotted it returns Ø. @TEST takes two coordinates. These are the X and Y coordinates of the pixel you want to test.

The @DRAW command is used to design a shape that you want displayed on the screen. But the shape will not actually appear until you use the @ROT command given below. @DRAW takes four parameters—the first of which is a string.

The string parameter actually contains the design information for the shape. The string should contain a series of digits between Ø and 9 in quotation marks. Each digit is an instruction on how to build up the shape.

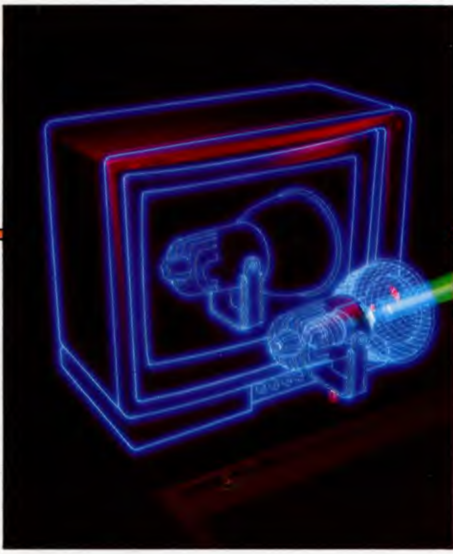
The design starts from the X and Y position given by parameters two and three and is plotted in the plot type specified by the fourth parameter. A Ø then moves one pixel to the right. A 1 moves one pixel up. A 2 moves one pixel down and a 3 moves one pixel to the left. In none of these cases is anything plotted.

A 5 moves one pixel to the right *and* plots a dot. A 6 moves one pixel up and plots a dot. A 7 moves one pixel down and plots a dot and an 8 moves one pixel to the left and plots a dot.

A 9 tells the @DRAW command to stop drawing.

INPUT's @DRAW command will only take 88 elements in the string parameter. So in the skier program on page 188, you have to use two @DRAW commands instead of one. The second one should start at the place the first one leaves off.

The @ROT command not only displays



- MAKING SURE OF SYNTAX
- CIRCLING
- ARCING
- ANGLING
- PAINTING
- FLASHING
- TESTING
- DRAWING
- ROTATING
- ADDING TEXT

what has been designed with the @DRAW command, it also rotates it through a given angle and draws it at a specified size. It takes two parameters.

The first is a number between Ø and 7 which specifies the angle of rotation from the verticle in multiples of 45 degrees. The second specifies the magnification. A 1 will draw the figure up at the same size as specified in the @DRAW. A 2 doubles the size—and so on. The @CHAR command prints a text character up on the screen at a specified size. It takes five parameters.

The first two are the X and Y coordinates of the screen position you want the character to appear at. The third is the screen code of the character—you'll find a full list of the screen codes in your User's Guide. The fourth is the plot type and the fifth specifies the size. This can be any number between 1 and 8, but it only magnifies the height of the letter by the amount given. The width stays the same.

The @TEXT command allows you to write text on the graphics screen. It takes six parameters.

The first two are the X and Y coordinates of the starting point of the text. The third is a string parameter which contains the text you want printed up between quotes. Pressing the CTRL and A keys at the beginning of the text will give capital letters when the text is displayed on the graphics screen. And pressing the CTRL and B keys will give ordinary, lower case letters. The screen code is not the same as the ASC( ) of the character.

The fourth parameter is the plot type. The fifth is the height of the letters. And the sixth specifies the number of pixels to be left blank between each letter.

THE ROUND TABLE

In this part of the hi-res graphics program there are a number of commands that deal with circles and ellipses. To draw circular shapes the computer needs to work out sines and cosines. As the ROM routine that does this takes rather a long time it is better to generate a table that the program can look up when it needs it.

This is done in BASIC. But the data table created is in RAM. Along with several of the routines in this part of the Commodore Hi-Res graphics article, it is located outside the protected area from \$CØØØ to \$CFFF.

BASIC PROGRAM

The first thing to do is to enter and RUN the following BASIC program:

```
2Ø sr=57.2957795
3Ø forf=Øto89
4Ø Poke47616+f,sin((f+.Ø83)/sr)*255.9
42 Poke477Ø6+f,sin((f+.916)/sr)*255.9
45 Poke47872+f,sin((f+.25Ø)/sr)*255.9
5Ø Poke47962+f,sin((f+.75Ø)/sr)*255.9
6Ø Poke48128+f,sin((f+.416)/sr)*255.9
7Ø Poke48218+f,sin((f+.583)/sr)*255.9
8Ø next
```

When you RUN this program, it POKEs a table of sines and cosines into memory. That done, you can NEW.

Because this table is located in the RAM under the BASIC interpreter itself you cannot use BASIC to save it, and we have provided a special machine-code save. After you have run the round table you can save the table (and all the machine code up to \$cfff), to disk with our routine with SYS 529Ø8 or to tape with SYS 529Ø1. This routine is typed in with your monitor as follows:

```
ceaØ 1Ø c3 4c dØ 41 a2 Ø1 dØ
cea8 Ø5 ff ff ff a2 Ø8 a9 Ø1
cebØ aØ Ø1 2Ø ba ff a5 Ø1 29
ceb8 fe 85 Ø1 a9 Ø5 a2 df aØ
cecØ ce 2Ø bd ff a9 ØØ 85 fb
cec8 a9 ba 85 fc a2 ff aØ cf
cedØ a9 fb 2Ø d8 ff a5 Ø1 Ø9
ced8 Ø1 85 Ø1 2Ø cc ff 6Ø 48
ceeØ 49 52 45 53 ØØ ØØ ØØ Ø2
ready.
```

The saved file is called "HIRES1".

THE MACHINE CODE

Using your machine code monitor (page 28Ø) type in the two blocks of code below; the first can be saved by SYS 529Ø1 or SYS 529Ø8, and the second must be saved using the monitor. This typing is an arduous task that can be done in stages. You can test each command as soon as you have typed in the necessary part. The symbolic listing is in 13 modules and a preface. The first section recognizes the commands from BASIC and each subsequent section completes one or more commands.

If you have purchased a professional assembler you can alternatively enter and assemble the symbolic source directly. You can assemble the sections independently provided you incorporate a copy of the preface with each (it's called modular programming). The source published here does assemble into the hexadecimal code published here. Unfortunately this powerful program requires the full Motorola standard assemble language to define, and the sections are too large to fit into the *INPUT* assembler, so they cannot be assembled on your *INPUT* assembler.

If you want to experiment with additional facilities to your HIRES routines, e.g. to move the screen or program to fit in with other software, or to construct several high resolution screens, the symbolic source has been carefully designed to be as clear as possible and the odd comment (prefaced by "!"), has been retained in the modules.

TESTING THE PROGRAM

When testing the program, it is helpful to (i) Start it by SYS 49152 rather than SYS 52ØØØ—you sacrifice the copyright message but nothing else, (ii) Test one command at a time, (iii) If the machine stops with a row of coloured squares on the graphic screen, type in @NRM RETURN (iv) Testing the other commands omitting to do @HIRES/@MULTI is a way of viewing error messages from the HIRES program, (v) You can hit RUN/STOP/RESTORE without destroying any BASIC programs or variables and then PEEK the data held in your machine code program

to see what went wrong. Finally, get rid of the monitor and type in:

SYS 52000

to switch on the hi-res commands properly.

You can then use most of the Commodore 64 graphics programs in *INPUT* without using Simons' BASIC. Whenever you want to reload the hi-res routines, remember to load both sections.

To SAVE anything that you have drawn on the screen do the following POKEs:

POKE43,0:POKE44,32:POKE45,0:POKE46,64

Then SAVE to tape with:

SAVE "filename",1,1

Because *INPUT*'s hi-res graphics program occupies different areas of memory from Simons' BASIC, you may experience some difficulty with routines that save more than one screen within the program itself.

This applies to the paged graphics programs on page 1134 and 1135, and the room designer program on pages 1269 to 1275 and 1308 to 1313.

```
49152 78 a9 0d 8d 08 03 a9 c0
49160 8d 09 03 58 60 20 73 00
49168 c9 40 f0 03 4c e7 a7 20
49176 73 00 a6 7a 86 fb a6 7b
49184 86 fc a0 00 a2 00 dd 00
49192 cf d0 18 e8 a9 0d dd 00
49200 cf f0 29 20 73 00 dd 00
49208 cf f0 f0 a5 fb 85 7a a5
49216 fc 85 7b a9 0d dd 00 cf
49224 f0 03 e8 d0 f6 e8 c8 c8
49232 20 79 00 c0 2c d0 cf a2
49240 0b 6c 00 03 b9 73 cf 85
49248 fb b9 74 cf 85 fc 6c fb
49256 00 00 00 00 00 00 00 00
49264 20 9b b7 e0 10 b0 0c 60
49272 20 fd ae 20 9e b7 e0 10
49280 b0 01 60 a2 0e 6c 00 03
49288 20 70 c0 8e 20 d0 20 78
49296 c0 8e 21 d0 4c ae a7 ff
49304 20 70 c0 8a 0a 0a 0a 0a
49312 85 02 20 78 c0 8a 05 02
49320 85 02 a9 20 85 fe a9 00
49328 85 fd a0 00 91 fd c8 c0
49336 00 d0 f9 e6 fe a6 fe e0
49344 40 d0 ef a9 3b 8d 11 d0
49352 ad 18 d0 29 f0 09 08 8d
49360 18 d0 a2 00 bd 00 04 9d
49368 28 a0 bd 00 05 9d 28 a1
49376 bd 00 06 9d 28 a2 bd 00
49384 07 9d 28 a3 e8 d0 e5 a5
49392 20 a0 00 99 00 04 99 fa
49400 04 99 f4 05 99 e8 06 c8
49408 d0 f1 a9 c8 8d 16 d0 4c
49416 34 c1 ff ff ff ff ff ff
49424 20 70 c0 8e f1 cf 20 78
49432 c0 8e f0 cf 20 78 c0 8e
49440 f2 cf a9 d8 8d 16 d0 ad
49448 f0 cf 0a 0a 0a 0a 0d f1
49456 cf 8d f3 cf a9 ff 8d f6
49464 cf 8d f7 cf 8d f8 cf 8d
49472 f9 cf a5 02 8d 21 d0 4c
49480 ae a7 00 00 00 00 00 00
49488 20 73 00 4c 34 c1 00 00
49496 00 00 00 00 00 00 00 00
49504 20 70 c0 8e f6 cf 20 78
49512 c0 8e f7 cf 20 78 c0 8e
49520 f8 cf ad f7 cf 0a 0a 0a
49528 0a 0d f6 cf 8d f9 cf 4c
49536 ae a7 ff ff ff ff ff ff
49544 20 73 00 a9 15 8d 18 d0
49552 a9 9b 8d 11 d0 a9 c8 8d
49560 16 d0 a5 01 29 fe 85 01
49568 a2 00 bd 00 04 9d 28 a8
49576 bd 00 05 9d 28 a9 bd 00
49584 06 9d 28 aa bd 00 07 9d
49592 28 ab e8 d0 eb a2 00 bd
49600 28 a0 9d 00 04 bd 28 a1
49608 9d 00 05 bd 28 a2 9d 00
49616 06 bd 28 a3 9d 00 07 e8
49624 d0 e5 a2 00 bd 00 d8 9d
49632 28 a4 bd 00 d9 9d 28 a5
49640 bd 00 da 9d 28 a6 bd 00
49648 db 9d 28 a7 e8 d0 e5 4c
49656 7a c2 00 00 82 74 00 ff
49664 20 9b b7 8a c9 00 f0 0d
49672 c9 01 f0 10 c9 02 f0 1b
49680 a6 0b 6c 00 03 a9 15 8d
49688 18 d0 d0 02 a9 17 8d 18
49696 d0 a5 20 2c 11 d0 f0 58
49704 4c 90 c1 a9 3b 8d 11 d0
49712 ad 18 d0 29 f0 09 08 8d
49720 18 d0 a5 01 29 fe 85 01
49728 a2 00 bd 28 a4 9d 00 d8
49736 bd 28 a5 9d 00 d9 bd 28
49744 a6 9d 00 da bd 28 a7 9d
49752 00 db e8 d0 e5 a2 00 bd
49760 28 a8 9d 00 04 bd 28 a9
49768 9d 00 05 bd 28 aa 9d 00
49776 06 bd 28 ab 9d 00 07 e8
49784 d0 e5 a5 01 09 01 85 01
49792 4c ae a7 ff ff ff ff ff
49800 a9 10 2c 16 d0 f0 06 0e
49808 e0 cf 2e e1 cf ad e0 cf
49816 29 07 49 07 8d ef cf ad
49824 e2 cf 29 07 0a 0a 85 fd
49832 ad e1 cf 4a ad e0 cf 6a
49840 4a 4a 85 fb ad e2 cf 4a
49848 4a 4a 85 fc 4a 66 fd 4a
49856 66 fd 18 65 fc 69 20 85
49864 fe a5 fb 0a 0a 0a 90 03
49872 e6 fe 18 65 fd 85 fd 90
49880 02 e6 fe 60 00 00 00 00
49888 20 73 00 4c eb c2 00 02
49896 20 fd ae 20 8a ad 20 f7
49904 b7 a5 15 30 30 c9 01 90
49912 08 d0 2a a5 14 c9 40 b0
49920 24 20 fd ae 20 9e b7 e0
49928 c9 b0 1a 60 ff ff ff ff
49936 20 fd ae 20 9e b7 e0 03
49944 90 12 a9 10 2c 16 d0 f0
49952 04 e0 05 90 07 68 68 a6
49960 0e 6c 00 03 8e e3 cf 60
49968 ff ff ff ff ff ff ff ff
49976 a9 00 8d ff cf 20 e0 c2
49984 a5 14 8d e0 cf a5 15 8d
49992 e1 cf 8e e2 cf 20 10 c3
50000 4c 58 c3 00 00 00 00 00
50008 20 88 c2 ad e2 cf c9 c9
50016 90 03 4c 53 c4 ad e1 cf
50024 30 0d c9 01 90 0c d0 07
50032 ad e0 cf c9 40 90 03 4c
50040 53 c4 a9 10 2c 16 d0 d0
50048 38 ad e3 cf c9 00 d0 0f
50056 a0 00 ae ef cf b1 fd 3d
50064 a8 cf 91 fd 4c 06 c4 c9
50072 02 d0 0f a0 00 ae ef cf
50080 b1 fd 5d a0 cf 91 fd 4c
50088 06 c4 a0 00 ae ef cf b1
50096 fd 1d a0 cf 91 fd 4c 06
50104 c4 ad ef cf 4a 8d ef cf
50112 ad e3 cf c9 00 d0 06 20
50120 5c c4 4c 06 c4 c9 04 d0
50128 0f a0 00 ae ef cf b1 fd
50136 5d b4 cf 91 fd 4c 06 c4
50144 c9 01 d0 0b 20 5c c4 1d
50152 b8 cf 91 fd 4c 06 c4 c9
50160 02 d0 0b 20 5c c4 1d bc
50168 cf 91 fd 4c 06 c4 20 5c
50176 c4 1d c0 cf 91 fd a5 fe
50184 49 20 85 fe 46 fe 66 fd
50192 46 fe 66 fd 46 fe 66 fd
50200 a9 04 45 fe 85 fe a0 00
50208 ad f9 cf c9 ff d0 1a a9
50216 10 2c 16 d0 f0 25 ad f3
50224 cf 91 fd 18 a5 fe 69 d4
50232 85 fe ad f2 cf 91 fd 10
50240 12 ea ad f9 cf 91 fd 18
50248 a5 fe 69 d4 85 fe ad f8
50256 cf 91 fd ad ff cf f0 01
50264 60 4c ae a7 a0 00 ae ef
50272 cf b1 fd 3d b0 cf 91 fd
50280 60 00 00 00 00 00 00 00
50288 a9 01 8d ff cf 20 e0 c2
50296 a5 14 8d e0 cf a5 15 8d
50304 e1 cf 8e e2 cf 20 e8 c2
50312 a5 14 8d d0 cf a5 15 8d
50320 d1 cf 8e d2 cf 20 10 c3
50328 ad d0 cf 38 ed e0 ef 8d
50336 3c 03 ad d1 cf ed e1 cf
50344 8d 3d 03 b0 0b a9 ff 8d
50352 3f 03 8d 4b 03 4c c2 c4
50360 a9 01 8d 3f 03 a9 00 8d
50368 4b 03 ad d2 cf 38 ed e2
50376 cf 8d 3e 03 b0 08 a9 ff
50384 8d 40 03 4c db c4 a9 01
50392 8d 40 03 ad 3f 03 8d 41
```

```
50400 Ø3 ad 4b Ø3 8d 4c Ø3 a9
50408 ØØ 8d 42 Ø3 ad 3f Ø3 c9
50416 ff fØ Øf ad 3c Ø3 8d 43
50424 Ø3 ad 3d Ø3 8d 44 Ø3 4c
50432 15 c5 ad eØ cf 38 ed dØ
50440 cf 8d 43 Ø3 ad e1 cf ed
50448 d1 cf 8d 44 Ø3 ad 4Ø Ø3
50456 c9 ff fØ Øe ad 3e Ø3 8d
50464 45 Ø3 a9 ØØ 8d 46 Ø3 4c
50472 39 c5 ad e2 cf 38 ed d2
50480 cf 8d 45 Ø3 a9 ØØ 8d 46
50488 Ø3 ad 43 Ø3 38 ed 45 Ø3
50496 ad 44 Ø3 ed 46 Ø3 bØ 26
50504 a9 ØØ 8d 41 Ø3 8d 4c Ø3
50512 ad 4Ø Ø3 8d 42 Ø3 ad 43
50520 Ø3 ae 45 Ø3 8d 45 Ø3 8e
50528 43 Ø3 ad 44 Ø3 ae 46 Ø3
50536 8e 44 Ø3 8d 46 Ø3 ad 44
50544 Ø3 6a ad 43 Ø3 6a 8d 47
50552 Ø3 a9 ØØ 8d 49 Ø3 8d 4a
50560 Ø3 8d 48 Ø3 ad eØ cf 8d
50568 dØ cf ad e1 cf 8d d1 cf
50576 ad e2 cf 8d d2 cf ad dØ
50584 cf 8d eØ cf ad d1 cf 8d
50592 e1 cf ad d2 cf 8d e2 cf
50600 2Ø 58 c3 ad 47 Ø3 18 6d
50608 45 Ø3 8d 47 Ø3 ad 48 Ø3
50616 6d 46 Ø3 8d 48 Ø3 ad 47
50624 Ø3 38 ed 43 Ø3 ad 48 Ø3
50632 ed 44 Ø3 9Ø 33 ad 47 Ø3
50640 38 ed 43 Ø3 8d 47 Ø3 ad
50648 48 Ø3 ed 44 Ø3 8d 48 Ø3
50656 ad dØ cf 18 6d 3f Ø3 8d
50664 dØ cf ad d1 cf 6d 4b Ø3
50672 8d d1 cf ad d2 cf 18 6d
50680 4Ø Ø3 8d d2 cf 4c 1d c6
50688 ad dØ cf 18 6d 41 Ø3 8d
50696 dØ cf ad d1 cf 6d 4c Ø3
50704 8d d1 cf ad d2 cf 18 6d
50712 42 Ø3 8d d2 cf ee 49 Ø3
50720 dØ Ø3 ee 4a Ø3 ad 4a Ø3
50728 cd 44 Ø3 bØ Ø3 4c 96 c5
50736 ad 49 Ø3 cd 43 Ø3 bØ Ø3
50744 4c 96 c5 ad ff cf c9 Ø1
50752 dØ Ø3 4c ae a7 6Ø ØØ ØØ
50760 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
50768 a9 Ø2 8d ff cf 2Ø eØ c2
50776 a5 14 8d bb c6 a5 15 8d
50784 bc c6 8e bd c6 2Ø e8 c2
50792 a5 14 8d be c6 a5 15 8d
50800 bf c6 8e cØ c6 2Ø 1Ø c3
50808 ad cØ c6 38 ed bd c6 fØ
50816 37 9Ø 35 ad bd c6 8d c1
50824 c6 ad bb c6 8d eØ cf ad
50832 bc c6 8d e1 cf ad be c6
50840 8d dØ cf ad bf c6 8d d1
50848 cf ad c1 c6 8d d2 cf 8d
50856 e2 cf cd cØ c6 fØ Ø9 ee
50864 c1 c6 2Ø 98 c4 4c 89 c6
50872 4c ae a7 ØØ ØØ ØØ ØØ ØØ
50880 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
50888 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
50896 a9 Ø1 8d ff cf 2Ø eØ c2
50904 a5 14 8d a9 c7 a5 15 8d
50912 aa c7 8e ad c7 2Ø fd ae
50920 2Ø 9e b7 8e af c7 2Ø 1Ø
50928 c3 2Ø 78 cØ 8a fØ Ø6 8d
50936 ae c7 2Ø ØØ c7 4c ae a7
50944 ad af c7 Øa Øa Øa 8d 27
50952 c7 ad af c7 4a 4a 4a 4a
50960 4a Ø9 dØ 8d 28 c7 a2 ØØ
50968 ad Øe dc 29 fe 8d Øe dc
50976 a5 Ø1 29 fb 85 Ø1 bd ff
50984 ff 9d a7 Ø2 e8 eØ Ø8 dØ
50992 f5 a5 Ø1 Ø9 Ø4 85 Ø1 ee
51000 Øe dc a2 ØØ 8e a7 c7 bd
51008 a7 Ø2 2Ø 4e c7 ae a7 c7
51016 e8 eØ Ø8 dØ ef 6Ø 8d a6
51024 c7 ae ae c7 8e a5 c7 ad
51032 a9 c7 8d ab c7 8d eØ cf
51040 ad aa c7 8d ac c7 8d e1
51048 cf aØ Ø8 8c a8 c7 ad ad
51056 c7 8d e2 cf ad a6 c7 39
51064 9f cf fØ Ø3 2Ø 58 c3 ad
51072 ab c7 18 69 Ø1 8d ab c7
51080 8d eØ cf ad ac c7 69 ØØ
51088 8d ac c7 8d e1 cf ac a8
51096 c7 88 dØ cf ee ad c7 ce
51104 a5 c7 dØ b3 6Ø ØØ ØØ ØØ
51112 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
51120 a9 Ø1 8d ff cf 2Ø eØ c2
51128 a5 14 8d a9 c7 a5 15 8d
51136 aa c7 8e 97 c8 a9 ff 85
51144 Ød 85 Øe a9 6a a2 ØØ 85
51152 49 86 4a 8e 9a c8 2Ø fd
51160 ae 2Ø b1 a9 a5 6a 8d 9b
51168 c8 a5 6b 8d 12 c8 a5 6c
51176 8d 13 c8 2Ø 1Ø c3 2Ø 78
51184 cØ 8a fØ Øc 8d ae c7 2Ø
51192 78 cØ 8e 98 c8 4c Ø6 c8
51200 2Ø 9e b7 4c ae a7 ae 9b
51208 c8 dØ Ø3 4c 94 c8 ae 9a
51216 c8 bd ff ff 8d 9c c8 c9
51224 12 fØ 13 c9 92 fØ 17 c9
51232 2Ø bØ 2b c9 Ø1 fØ 17 c9
51240 Ø2 fØ 1b 4c 8Ø c8 a9 8Ø
51248 8d 99 c8 4c 8Ø c8 a9 ØØ
51256 8d 99 c8 4c 8Ø c8 a9 dØ
51264 8d 12 c7 4c 8Ø c8 a9 d8
51272 8d 12 c7 4c 8Ø c8 29 8Ø
51280 4a 8d 9d c8 ad 9c c8 29
51288 3f Ød 9d c8 Ød 99 c8 8d
51296 af c7 8e 9a c8 ad 97 c8
51304 8d ad c7 2Ø ØØ c7 ad a9
51312 c7 18 6d 98 c8 8d a9 c7
51320 ad aa c7 69 ØØ 8d aa c7
51328 ae 9a c8 e8 ec 9b c8 fØ
51336 Ø6 8e 9a c8 4c Øe c8 a9
51344 dØ 8d 12 c7 4c ae a7 ØØ
51352 ØØ ØØ ØØ ØØ ØØ ØØ ff ff
51360 2Ø 73 ØØ a9 Ø1 8d ff cf
51368 a9 ff 85 Ød 85 Øe a9 6a
51376 a2 ØØ 85 49 86 4a 8e e8
51384 cf 2Ø b1 a9 a5 6a 8d e9
51392 cf a5 6b 8d fa c8 a5 6c
51400 8d fb c8 2Ø e8 c2 a5 14
51408 8d e4 cf a5 15 8d e5 cf
51416 8e e6 cf 2Ø 1Ø c3 aØ 8Ø
51424 8c eb cf 8c ec cf ac dc
51432 cf 8c ea cf a9 2c 8d Ø2
51440 ca ae e8 cf ec e9 cf 1Ø
51448 3Ø bd ff ff 8d e7 cf c9
51456 39 fØ 26 c9 3Ø fØ 25 c9
51464 31 fØ 4d c9 32 fØ Øe c9
51472 33 fØ Øe c9 35 fØ 15 c9
51480 36 fØ 3d c9 37 fØ 65 c9
51488 38 dØ Ø3 4c bØ c9 4c e1
51496 ce 4c ae a7 ad ec cf 18
51504 6d d5 cf 8d ec cf ad e4
51512 cf 6d ca cf 2Ø 13 ca 6d
51520 cb cf 8d e5 cf ad eb cf
51528 18 6d d9 cf 8d eb cf ad
51536 e6 cf 6d cf cf 4c d9 c9
51544 ad ec cf 18 6d d6 cf 8d
51552 ec cf ad e4 cf 6d c4 cf
51560 2Ø 13 ca 6d c5 cf 8d e5
51568 cf ad eb cf 18 6d da cf
51576 8d eb cf ad e6 cf 6d cc
51584 cf 4c d9 c9 ad ec cf 18
51592 6d d7 cf 8d ec cf ad e4
51600 cf 6d c6 cf 2Ø 13 ca 6d
51608 c7 cf 8d e5 cf ad eb cf
51616 18 6d db cf 8d eb cf ad
51624 e6 cf 6d cd cf 4c d9 c9
51632 ad ec cf 18 6d d4 cf 8d
51640 ec cf ad e4 cf 6d c8 cf
51648 2Ø 13 ca 6d c9 cf 8d e5
51656 cf ad eb cf 18 6d d8 cf
51664 8d eb cf ad e6 cf 6d ce
51672 cf cd e6 cf fØ Ø8 8d e6
51680 cf a9 2Ø 8d Ø2 ca ad e7
51688 cf c9 34 3Ø 18 ad e4 cf
51696 8d eØ cf ad e5 cf 8d e1
51704 cf ad e6 cf 8d e2 cf 8e
51712 e8 cf 2Ø 58 c3 ce ea cf
51720 fØ Ø3 4c ec c8 ee e8 cf
51728 4c e6 c8 Ø8 cd e4 cf fØ
51736 Øa 48 a9 2Ø 8d Ø2 ca 68
51744 8d e4 cf ad e5 cf 28 6Ø
51752 ff ff ff ff ff ff ff ff
51760 a9 Ø2 8d ff cf 2Ø eØ c2
51768 a5 14 8d e4 cf a5 15 8d
51776 e5 cf 8e e6 cf 2Ø e8 c2
51784 a5 14 8d e7 cf a5 15 8d
51792 e8 cf 8e e9 cf 2Ø 1Ø c3
51800 ad e4 cf 18 6d e7 cf 8d
51808 ea cf ad e5 cf 6d e8 cf
51816 8d eb cf c9 Ø1 3Ø Ø9 dØ
51824 17 ad ea cf c9 4Ø 1Ø 1Ø
51832 ad e6 cf 18 6d e9 cf 8d
51840 ec cf bØ Ø4 c9 c9 9Ø Ø5
51848 a2 Øb 6c ØØ Ø3 ad e4 cf
51856 8d eØ cf ad e5 cf 8d e1
51864 cf ad e6 cf 8d e2 cf 8d
51872 d2 cf ad ea cf 8d dØ cf
51880 ad eb cf 8d d1 cf 2Ø 98
51888 c4 ad ea cf 8d eØ cf 8d
51896 dØ cf ad eb cf 8d e1 cf
51904 8d d1 cf ad ec cf 8d d2
51912 cf ad e6 cf 8d e2 cf 2Ø
51920 98 c4 ad ec cf 8d e2 cf
51928 8d d2 cf ad e4 cf 8d dØ
51936 cf ad e5 cf 8d d1 cf ad
51944 ea cf 8d eØ cf ad eb cf
51952 8d e1 cf 2Ø 98 c4 ad e6
51960 cf 8d d2 cf ad ec cf 8d
51968 e2 cf ad e4 cf 8d eØ cf
51976 8d dØ cf ad e5 cf 8d e1
51984 cf 8d d1 cf 2Ø 98 c4 4c
51992 ae a7 ff ff ff ff ff ff
52000 a9 ff 85 33 85 37 a9 1f
52008 85 34 85 38 2Ø 18 e5 a9
52016 Ø6 8d 2Ø dØ a9 Øf 8d 21
52024 dØ a2 ØØ bd 5d cb 2Ø d2
52032 ff e8 eØ 2e dØ f5 a5 37
52040 38 e5 2d aa a5 38 e5 2e
52048 2Ø cd bd a9 6Ø aØ e4 2Ø
52056 1e ab 4c ØØ cØ 1f 12 2Ø
52064 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
52072 49 4e 5Ø 55 54 2Ø 48 49
52080 52 45 53 2Ø 47 52 41 5Ø
52088 48 49 43 53 2Ø 2Ø 2Ø 2Ø
52096 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø 2Ø
52104 2Ø 2Ø 2Ø ff ff ff ff ff
52112 2Ø 9b b7 eØ Ø8 1Ø Ø8 8e
52120 ff cf 8a Øa aa 1Ø Ø5 a9
52128 Øb 6c ØØ Ø3 bd 31 cc 8d
52136 c4 cf bd 32 cc 8d c5 cf
52144 bd 3d cc 8d c8 cf bd 3e
52152 cc 8d c9 cf bd 35 cc 8d
52160 ca cf bd 36 cc 8d cb cf
52168 bd 39 cc 8d c6 cf bd 3a
52176 cc 8d c7 cf ae ff cf bd
52184 4d cc 8d cc cf bd 51 cc
52192 8d cd cf bd 53 cc 8d ce
52200 cf bd 4f cc 8d cf cf bd
52208 5b cc 8d d4 cf bd 5f cc
52216 8d d5 cf bd 5d cc 8d d6
52224 cf bd 61 cc 8d d7 cf bd
52232 61 cc 8d d8 cf bd 5d cc
52240 8d d9 cf bd 5b cc 8d da
52248 cf bd 5f cc 8d db cf 2Ø
52256 fd ae 2Ø 9e b7 8e dc cf
52264 8a dØ Ø3 4c 9f cb 4c ae
52272 a7 ØØ ØØ ØØ ØØ Ø1 ØØ ØØ
52280 ØØ ØØ ØØ ff ff ff ff ff
52288 ff ØØ ØØ ØØ ØØ Ø1 ØØ ØØ
52296 ØØ ØØ ØØ ff ff ff ff ØØ
52304 ØØ Ø1 ØØ ØØ ff ff ff ØØ
52312 ØØ Ø1 ØØ ØØ 4b ØØ b6 ØØ
52320 b6 ØØ 4b ØØ 4b ØØ b6 ØØ
52328 b6 ØØ ØØ ØØ ØØ ØØ ØØ ØØ
52336 2Ø eØ c2 a5 14 8d eØ cf
52344 a5 15 8d e1 cf 8e e2 cf
```

```
52352 20 88 cc 4c ae a7 ff ff
52360 20 88 c2 a0 00 ad 16 d0
52368 29 10 d0 0f ae ef cf bd
52376 a0 cf 31 fd f0 02 a9 01
52384 85 02 60 ad ef cf 4a aa
52392 a0 00 bd c0 cf d0 eb ad
52400 1e cd 18 69 01 69 00 8d
52408 1e cd cd 1f cd d0 5b a9
52416 00 8d 1e cd a5 fb 48 a5
52424 fc 48 a5 fd 48 a5 fe 48
52432 98 48 a9 7f 8d 0d dc a9
52440 00 85 fb 85 fd a9 04 85
52448 fc a9 d8 85 fe a0 00 b1
52456 fd 29 0f cd 1d cd d0 06
52464 b1 fb 49 80 91 fb c8 d0
52472 04 e6 fc e6 fe c0 e8 d0
52480 e6 a5 fc c9 07 d0 e0 68
52488 a8 68 85 fe 68 85 fd 68
52496 a5 fc 68 85 fb a9 81 8d
52504 0d dc 4c 31 ea ff 01 00
52512 20 70 c0 8e 1d cd 20 fd
52520 ae 20 9e b7 8e 1f cd a9
52528 00 8d 1e cd ad 14 03 c9
52536 af f0 14 a9 7f 8d 0d dc
52544 a9 af 8d 14 03 a9 cc 8d
52552 15 03 a9 81 8d 0d dc 4c
52560 ae a7 00 00 00 00 00 00
52568 00 00 00 00 00 00 00 00
52576 20 73 00 a9 00 8d 1f cd
52584 4c ae a7 00 00 00 00 00
52592 20 d5 cd 20 eb cd 20 23
52600 ce f0 06 20 00 40 4c ae
52608 a7 6c 00 03 ff ff ff ff
52616 ff ff ff ff ff ff ff ff
52624 20 d5 cd 20 ff cd 4c 76
52632 cd ff ff ff ff ff ff ff
52640 20 d5 cd 20 3f ce a5 14
52648 8d b3 41 8d b0 41 a5 15
52656 8d b4 41 20 23 ce f0 c9
52664 ad d0 cf 8d e0 cf ad d1
52672 cf 8d e1 cf ad d2 cf 8d
52680 e2 cf 20 58 c3 a9 2c 20
52688 4f 40 4c 98 c4 a9 01 8d
52696 ff cf 20 e0 c2 a5 15 8d
52704 d0 cf a5 15 8d d1 cf 8e
52712 d2 cf 60 a9 00 8d b3 41
52720 8d b4 41 8d 3f 40 8d 4b
52728 40 a9 01 8d 23 40 60 20
52736 3f ce a5 14 8d b3 41 a5
52744 15 8d b4 41 20 3f ce a5
52752 14 8d 3f 40 a5 15 8d 4b
52760 40 20 fd ae 20 9e b7 8e
52768 23 40 60 20 e8 c2 a5 14
52776 8d ad 41 a5 15 8d ae 41
52784 8e af 41 20 10 c3 ad ad
52792 41 d0 03 ad af 41 60 20
52800 fd ae 20 8a ad 20 f7 b7
52808 a5 15 c9 00 f0 06 a5 14
52816 c9 68 10 01 60 a5 14 38
52824 e9 68 85 14 a5 15 e9 01
52832 85 15 d0 e4 60 00 00 02
52840 00 00 00 00 00 00 00 00
52848 00 00 00 0a 00 00 00 af
52856 00 00 00 00 82 74 00 ff
52864 a9 01 8d ff cf a2 01 8e
52872 1b 43 a2 ff 8e 00 bf 20
52880 e0 c2 a5 14 8d 18 43 a5
52888 15 8d 19 43 8e 1a 43 20
52896 10 c3 4c d0 41 ff ff ff
52904 ff ff ff ff ff ff ff ff
52912 ff ff ff ff ff ff ff ff
52920 ff ff ff ff fe ff ff ff
52928 00 00 00 00 00 00 00 00
52936 00 00 00 00 00 00 00 00
52944 00 00 00 00 00 00 00 00
52952 00 00 00 00 00 00 00 00
52960 00 00 00 00 00 00 00 02
52968 00 00 00 00 00 00 00 00
52976 00 00 00 02 00 00 00 bf
52984 00 00 00 00 82 74 00 ff
52992 43 4f 4c 4f 55 52 0d 48
53000 49 52 45 53 0d 4d 55 4c
53008 54 49 0d 4e 52 4d 0d 4c
53016 4f 57 43 4f 4c 0d 48 49
53024 43 4f 4c 0d 50 4c 4f 54
53032 0d 4c 49 4e 45 0d 42 4c
53040 4f 43 4b 0d 50 41 b5 0d
53048 54 45 53 54 0d 43 53 45
53056 54 0d 52 45 43 0d 43 48
53064 41 52 0d 54 45 58 54 0d
53072 41 52 43 0d 41 4e 47 4c
53080 0d 43 49 52 43 4c 45 0d
53088 44 52 41 57 0d 52 4f 54
53096 0d 46 4c 41 53 48 0d 4f
53104 46 46 0d 88 c0 98 c0 10
53112 c1 88 c1 60 c1 50 c1 38
53120 c3 70 c4 50 c6 80 ce 70
53128 cc 00 c2 30 ca d0 c6 b0
53136 c7 90 cd a0 cd 70 cd a0
53144 c8 90 cb 20 cd 60 cd 20
53152 01 02 04 08 10 20 40 80
53160 fe fd fb f7 ef df bf 7f
53168 fc f3 cf 3f 03 0c 30 c0
53176 02 08 20 80 01 04 10 40
53184 03 0c 30 c0 1d 43 00 00
53192 01 08 03 00 00 af 00 01
53200 63 a1 66 a4 01 37 00 72
53208 d1 03 39 06 00 00 00 00
53216 00 00 00 00 00 00 00 02
53224 00 00 00 00 00 00 00 00
53232 00 00 00 02 00 00 00 af
53240 00 00 00 64 00 0a 00 00
16384 ad b3 41 8d b0 41 a9 20
16392 20 4f 40 20 ac 40 a9 bb
16400 20 a6 40 20 ac 40 a9 bc
16408 20 a6 40 20 ac 40 ad b3
16416 41 18 69 ff 8d b3 41 90
16424 03 ee b4 41 ae b4 41 f0
16432 0c c9 68 90 08 e9 68 ce
16440 b4 41 8d b3 41 38 e9 ff
16448 90 be ed 23 40 b0 b9 ad
16456 b4 41 c9 ff d0 b2 60 8d
16464 52 41 ad b4 41 d0 2f ad
16472 b0 41 c9 5a b0 06 a2 00
16480 a0 01 d0 3a c9 b4 b0 0f
16488 a9 0e 18 ed b0 41 8d b0
16496 41 a2 00 a0 00 f0 27 a2
16504 01 ad b0 41 38 e9 b4 8d
16512 b0 41 a0 00 f0 18 ad b0
16520 41 38 e9 0e 30 e9 8d b0
16528 41 a9 b4 18 ed b0 41 8d
16536 b0 41 a2 01 a0 01 8e cb
16544 40 8c 1b 41 a9 ba 8d b7
16552 40 8d 08 41 ae b0 41 a5
16560 01 29 fe 85 01 bd 00 ff
16568 8d 85 41 ad ad 41 8d 91
16576 41 ad ae 41 8d 89 41 20
16584 7c 41 a9 ff d0 1b ad b1
16592 41 18 6d d0 cf a8 ad b2
16600 41 6d d1 cf f0 1b c9 02
16608 b0 47 c0 40 90 13 4c 68
16616 41 ad d0 cf 38 ed b1 41
16624 a8 ad d1 cf ed b2 41 90
16632 6f 8d e1 cf 8c e0 cf a9
16640 b4 18 ed b0 41 aa bd 00
16648 ff 8d 85 41 ad af 41 8d
16656 91 41 a9 00 8d 89 41 20
16664 7c 41 a9 ff d0 0f ad b1
16672 41 18 6d d2 cf b0 41 c9
16680 c8 b0 3d 90 09 ad d2 cf
16688 38 ed b1 41 90 32 8d e2
16696 cf 48 ac e0 cf 8c 5c 41
16704 a2 05 68 48 dd b5 41 d0
16712 06 98 dd bb 41 f0 06 ca
16720 10 f0 20 58 c3 a0 05 68
16728 99 b5 41 a9 ff 99 bb 41
16736 88 10 02 a0 05 8c 56 41
16744 ad b0 41 38 e9 5a b0 02
16752 69 b4 8d b0 41 a5 01 09
16760 01 85 01 60 a9 00 8d b1
16768 41 8d b2 41 a2 ff f0 24
16776 a0 01 f0 03 8e b1 41 18
16784 69 ff 90 09 18 ee b1 41
16792 d0 03 ee b2 41 ca d0 f0
16800 29 80 f0 08 ee b1 41 d0
16808 03 ee b2 41 60 00 00 00
16816 00 00 00 00 00 00 00 00
16824 00 00 00 00 00 00 00 00
16832 00 00 00 00 00 00 00 00
16840 00 00 00 00 00 00 00 00
16848 a5 01 29 fe 85 01 ee 1a
16856 43 20 67 42 ce 1a 43 ad
16864 18 43 d0 03 ce 19 43 ce
16872 18 43 20 67 42 ee 18 43
16880 d0 03 ee 19 43 ce 1a 43
16888 20 67 42 ee 1a 43 ee 18
16896 43 d0 03 ee 19 43 20 67
16904 42 ad 18 43 d0 03 ce 19
16912 43 ce 18 43 ad 19 43 8d
16920 e1 cf ad 18 43 8d e0 cf
16928 ad 1a 43 8d e2 cf 20 58
16936 c3 20 3c 42 ad 1a 43 c9
16944 ff d0 a3 a5 01 09 01 85
16952 01 4c ae a7 ae 1b 43 ca
16960 e0 00 d0 0d ad 1c 43 c9
16968 01 d0 06 a9 00 8d 1c 43
16976 ca bd 00 bd 8d 18 43 bd
16984 00 be 8d 19 43 bd 00 bf
16992 8d 1a 43 8e 1b 43 60 a9
17000 10 2c 16 d0 f0 10 a9 00
17008 8d e1 cf ad 18 43 8d e0
17016 cf c9 a0 90 1a 60 ad 19
17024 43 8d e1 cf f0 0b ad 18
17032 43 8d e0 cf c9 40 90 07
17040 60 ad 18 43 8d e0 cf ad
17048 1a 43 8d e2 cf c9 c8 90
17056 01 60 20 88 cc a5 02 f0
17064 01 60 a9 08 8d 1d 43 ae
17072 1b 43 ad 18 43 48 ac 1a
17080 43 c8 ca f0 3d ce 1d 43
17088 f0 38 98 38 fd 00 bf f0
17096 06 c9 02 90 13 d0 eb 68
17104 48 dd 00 bd d0 e4 ad 19
17112 43 fd 00 be d0 dc f0 28
17120 68 48 fd 00 bd f0 ef b0
17128 06 c9 fe 90 cd b0 e7 c9
17136 fe d0 c7 bd 00 be d0 10
17144 f0 c0 ae 1b 43 e8 d0 04
17152 e8 8e 1c 43 8e 1b 43 ca
17160 68 9d 00 bd ad 19 43 9d
17168 00 be 88 98 9d 00 bf 60
17176 00 00 00 00 00 00 00 00
17184 00 00 00 00 00 00 00 00
```

SAVE ROUTINE

```
setlfs = $ffba
setnam = $ffbd
save = $ffd8
clrchn = $ffcc
* = 52901
```

[N.B.: Programmers ref. guide p. 294 is wrong.]

```
tpsave ldx #1
        !device
   bne   saveme
* = 52908
dksave ldx #8
saveme lda #1
    ldy   #1
    jsr   setlfs
    lda   $01
    and   #%11111110
    sta   $01
    lda
        #endofn — hiresn
    ldx   # <hiresn
    ldy   # >hiresn
    jsr   setnam
    lda   # <sincos
    sta   frekzp
    lda   # >sincos
    sta   frekzp + 1
```

```
  ldx  #<exitpl
  ldy  #>exitpl
  lda  #frekzp
  jsr  save
  lda  $Ø1
  ora  #%ØØØØØØØ1
  sta  $Ø1
  jsr  clrchn
  rts
hiresn txt
    "hires"
endofn=*
```

PREFACE

```
! input hires
    basic
! standard
    Preface —
    definitions
!Part 1 —
    addresses of
    modules
welcom=$cb2Ø
modul 1=$cØØØ
colour=$cØ88
hires=$cØ98
multi=$c11Ø
nrm=$c188
lowcol=$c16Ø
hicol=$c15Ø
plot=$c338
line=$c47Ø
block=$c65Ø
paint=$ce8Ø
test=$cc7Ø
cset=$c2ØØ
rec=$ca3Ø
char=$6dØ
text=$c7bØ
arc=$cd9Ø
angl=$cdaØ
circle=$cd7Ø
draw=$c8aØ
rot=$cb9Ø
flash=$cd2Ø
off=$cd6Ø
!addresses of
    subsections
cgset1=$cØ7Ø
cget=$cØ78
cpixad = $c288
getxy1=$c2eØ
getxy=$c2e8
gptype=$c31Ø
bitmsk=$cfaØ
plotex=$c358
lineex=$c498
ctestp=$cc88
circ1Ø=$4ØØØ
pain2Ø=$41dØ
!part 2 —
    addresses of
    variables
somwer=$ØØØ2
    !work byte on
    zero page
argho=$ØØ6a
    !fac#2 4—byte
    work area
frekzp=$ØØfb
    !four bytes
    free core
holder = $cfe4
bscren=$aØ28
!... holds
    backup of
    low-res screen
!table used by
    @draw, set by
    @rot
  !.. contains
    unit vectors
    in 4
    directions
tbxpi=$cfc4
tbypix=$cfcc
tbxpar=$cfd4
tbypar=$cfd8
drsize=$cfdc
!tables used by
    @circle/@arc/
    @angl (3*18Ø
    bytes)
sincos=$baØØ
!stack used by
    @paint (3*256
    bytes)
pstack=$bdØØ.
!'there' is
    (xthere,ythere)
    — other end
    of line in
    @line
xthere=$cfdØ
xthrhi=$cfd1
ythere=$cfd2
!used by 'plot'
    routine ....
xplot = $cfeØ
xhigh = $cfe1
yplot=$cfe2
ptype=$cfe3
usebit=$cfef
ink1=$cff1
ink2=$cffØ
ink3=$cff2
ink12=$cff3
inkx1 = $cff6
inkx2 = $cff7
inkx3=$cff8
inkx12=$cff9
exitpl=$cfff
    !how to leave
    @plot s/r
basic=Ø
linejb = 1
rtsjob = 2
!part 3 —
    addresses of
    c64 rom
    routines
chrget=$ØØ73
ierror=$Ø3ØØ
infix1=$b79b
infix=$b79e
inposn=$aefd
xbasic=$a7ae
infacc=$ad8a
facfix=$b7f7
fixlow=$ØØ14
fixhi=$ØØ15
instrg=$a9b1
valtyp=$ØØØd
intflg=$ØØØe
vstrng=$ff
forpnt=$49
!part 4a— other
    c64 addresses
lobase=$Ø4ØØ
    !base of
    low-res screen
quartr=25Ø
attr1=lobase
    !base of
    attribute mem
attr2=attr1+quar
    tr
attr3=attr2+quar
    tr
attr4=attr3+quar
    tr
ch1bse=$dØØØ
    !set1
    character rom
ch2bse=$d8ØØ
    !set2
    character rom
vic=$dØØØ !vic
    chip base
cmemry=$d8ØØ
    !colour ram
    memory
ciachp = $dcØØ
ciamsk=ciachp+$Ø
    d !masks
    interrupts
    (poke it)
iset = %1ØØØØØØØ
!... with this
    bit on it says
    'set trap(s)'
iclear=Ø
!... with same
    bit off it
    says 'cancel
    trap(s)'
timera=%ØØØØØØØ1
ctimer=ciachp+$Ø
    e
timbit=%ØØØØØØØ1
hibase=$2ØØØ
    !base of hires
    screen
hitop=$4ØØØ !top
    of hires
    screen
!part 4b— vic
    chip registers
    and putative
    contents
viccty=vic+$11
scrol3=%Ø11
row25=8
notblk=16
bitmod=32
ignore=128
vicctx=vic+$16
row4Ø=8
multic=16
unused=%11ØØØØØØ
vicmem=vic+$18
!low-res
    screenpointer
    in thousands
    goes in the
    top of it
lorfld=%1111ØØØØ
lorup=>lobase
lorup4=lorup+lorup
    +lorup+lorup
lorptr=<lorup4
ch1up=>ch1bse
ch2up=>ch2bse
ch1up4=ch1up+ch1up
    +ch1up+ch1up
ch2up4=ch2up+ch2up
    +ch2up+ch2up
ch1u16=<ch1up4+<
    ch1up4+<ch1up4
    +<ch1up4
ch2u16=<ch2up4+<
    ch2up4+<ch2up4
    +<ch2up4
ch1u64=ch1u16+ch
    1u16+ch1u16+ch
    1u16
ch2u64=ch2u16+ch
    2u16+ch2u16+ch
    2u16
ch1ptr=>ch1u64
ch2ptr=>ch2u64
!hires pointer
    in thousands
    goes in the
    top of it
hiup=>hibase
hiup4=hiup+hiup+
    hiup+hiup
hiup16=<hiup4+
    <hiup4
    +<hiup4+
    <hiup4
hiup64=hiup16=hi
    up16+hiup16+hi
    up16
hirptr=>hiup64
onmem=%ØØØØØØØ1
border=vic+$2Ø
bkgrnd=vic+$21
!part 5 — things
    worth naming
serror=11
    !"syntax
    error"
orange = 14
    !"out-of-range"
nybble=%1111
ymax=2ØØ
xmax=32Ø
xcmax=16Ø
    !screen width
    (multic)
rowlen=4Ø
minus1=%11111111
plus1=1
```

MODULE 1

```
!tables
cr=13
int=181
*=52992 !=$cfØØ
keylst txt
    "colour"
  byt  cr
  txt  "hires"
  byt  cr
  txt  "multi"
  byt  cr
  txt  "nrm"
  byt  cr
  txt  "lowcol"
  byt  cr
  txt  "hicol"
  byt  cr
  txt  "plot"
  byt  cr
  txt  "line"
  byt  cr
  txt  "block"
  byt  cr
  byt  'p,'a,int
  byt  cr
  txt  "test"
  byt  cr
  txt  "cset"
  byt  cr
  txt  "rec"
  byt  cr
  txt  "char"
  byt  cr
  txt  "text"
  byt  cr
  txt  "arc"
  byt  cr
  txt  "angl"
  byt  cr
  txt  "circle"
  byt  cr
  txt  "draw"
  byt  cr
  txt  "rot"
  byt  cr
  txt  "flash"
  byt  cr
  txt  "off"
  byt  cr
jumplst wor
    colour
  wor  hires
  wor  multi
  wor  nrm
  wor  lowcol
  wor  hicol
  wor  plot
  wor  line
  wor  block
  wor  paint
  wor  test
  wor  cset
  wor  rec
  wor  char
  wor  text
  wor  arc
  wor  angl
  wor  circle
  wor  draw
  wor  rot
  wor  flash
  wor  off
keytot=*-jmplst
!program
txtbak=frekzp
jmpptr=frekzp
chragn=$ØØ79
txtptr=$ØØ7a
igone=$Ø3Ø8
ebasic=$a7e7
*=modul 1
  sei
  lda  #<examin
  sta  igone
  lda  #>examin
  sta  igone+1
```

```
    cli
    rts
examin jsr
      chrget
    cmp   #'@
    beq   exam02
    jmp   ebasic
exam02 jsr
      chrget
    ldx   txtptr
    stx   txtbak
    ldx   txtptr+1
    stx   txtbak+1
    ldy   #0
    ldx   #0
exam03 cmp
      keylst,x
    bne exam04
exam07 inx
    lda   #cr
    cmp   keylst,x
    beq   exam10
    jsr   chrget
    cmp   keylst,x
    beq   exam07
    lda   txtbak
    sta   txtptr
    lda   txtbak+1
    sta   txtptr+1
exam04 lda #cr
    cmp   keylst,x
    beq exam05
    inx
    bne   exam04
exam05 inx
    iny
    iny
    jsr   chragn
    cpy   #keytot
    bne   exam03
exam99 ldx
      #serror
    jmp   (ierror)
exam10 lda
      jmplst,y
    sta   jmpptr
    lda   jmplst+1,y
    sta   jmpptr+1
    jmp   (jmpptr)
```

MODULE 2

```
*=cget1
    jsr   infix1
    cpx   #nybble+1
    bcs   cerror
    rts
*=cget
    jsr   inposn
    jsr   infix
    cpx   #nybble+1
    bcs   cerror
    rts
cerror ldx
      #orange
    jmp   (ierror)
*=colour
    jsr   cget1
    stx   border
    jsr   cget
    stx   bkgrnd
    jmp   xbasic
scrptr=frekzp+2
*=hires
    jsr   cget1
    txa
    asl   a
    asl   a
    asl   a
    asl   a
    sta   somwer
    jsr   cget
    txa
    ora   somwer
    sta   somwer
    lda   #>hibase
    sta   scrptr+1
    lda   #<hibase
    sta   scrptr
zer256 ldy #0
zer1 sta
      (scrptr),y
    iny
    cpy   #0
    bne   zer1
    inc   scrptr+1
    ldx   scrptr+1
    cpx   #>hitop
    bne   zer256
    lda
      #scrol3+row25
      +notblk+bitmod
    sta   viccty
    lda   vicmem
    and   #lorfld
    ora   #hirptr
    sta   vicmem
    ldx   #0
hire05 lda
      lobase,x
    sta   bscren,x
    lda
      lobase+$100,x
    sta
      bscren+$100,x
    lda
      lobase+$200,x
    sta
      bscren+$200,x
    lda
      lobase+$300,x
    sta
      bscren+$300,x
    inx
    bne   hire05
    lda   somwer
    ldy   #0
copy1 sta
      attr1,y
    sta   attr2,y
    sta   attr3,y
    sta
      attr4+quartr-$
      100,y
    iny
    bne copy1
    lda
      #unused+row40
    sta   vicctx
    jmp   multi5
*=multi
    jsr   cget1
    stx   ink1
    jsr   cget
    stx   ink2
    jsr   cget
    stx   ink3
    lda
      #unused+multic
      +row40
    sta   vicctx
    lda   ink2
    asl   a
    asl   a
    asl   a
    asl   a
    ora   ink1
    sta   ink12
multi5 lda #$ff
    sta   inkx1
    sta   inkx2
    sta   inkx3
    sta   inkx12
    lda   somwer
    sta   bkgrnd
multi9 jmp
      xbasic
*=hicol
    jsr   chrget
    jmp   multi5
```

MODULE 3

```
*=lowcol
    jsr   cget1
    stx   inkx1
    jsr   cget
    stx   inkx2
    jsr   cget
    stx   inkx3
    lda   inkx2
    asl   a
    asl   a
    asl   a
    asl   a
    ora   inkx1
    sta   inkx12
    jmp   xbasic
*=nrm
    jsr   chrget
    lda
      #lorptr+ch1ptr
      +onmem
    sta   vicmem
nrm02 lda
      #scrol3+row25
      +notblk+ignore
    sta   viccty
    lda
      #unused+row40
    sta   vicctx
    lda   $01
    and   #%11111110
    sta   $01
    ldx   #0
    lda   attr1,x
    sta
      bscren+$800,x
nrm04 lda
      attr1+$100,x
    sta
      bscren+$900,x
    lda
      attr1+$200,x
    sta
      bscren+$a00,x
    lda
      attr1+$300,x
    sta
      bscren+$b00,x
    inx
    bne   nrm04
    ldx   #0
nrm07 lda
      bscren,x
    sta   lobase,x
    lda
      bscren+$100,x
    sta
      lobase+$100,x
    lda
      bscren+$200,x
    sta
      lobase+$200,x
    lda
      bscren+$300,x
    sta
      lobase+$300,x
    inx
    bne   nrm07
    ldx   #0
nrm10 lda
      cmemry,x
    sta
      bscren+$400,x
    lda
      cmemry+$100,x
    sta
      bscren+$500,x
    lda
      cmemry+$200,x
    sta
      bscren+$600,x
    lda
      cmemry+$300,x
    sta
      bscren+$700,x
    inx
    bne   nrm10
    jmp   cset30
*=cset
    jsr   infix1
    txa
    cmp   #0
    beq   cset0
    cmp   #1
    beq   cset1
    cmp   #2
    beq   cset2
    ldx   serror
    jmp   (ierror)
cset0 lda
      #lorptr+ch1ptr
      +onmem
    sta   vicmem
    bne   cset01
cset1 lda
      #lorptr+ch2ptr
      +onmem
cset01 sta
      vicmem
    lda   bitmod
    bit   viccty
    beq   zbasic
    jmp   nrm02
cset2 lda
      #scrol3+row25
      +notblk+bitmod
    sta   viccty
    lda   vicmem
    and   #lorfld
    ora   #hirptr
    sta   vicmem
    lda   $01
    and   #%11111110
    sta   $01
    ldx   #0
cset24 lda
      bscren+$400,x
    sta   cmemry,x
    lda
      bscren+$500,x
    sta
      cmemry+$100,x
    lda
      bscren+$600,x
    sta
      cmemry+$200,x
    lda
      bscren+$700,x
    sta
      cmemry+$300,x
    inx
    bne   cset24
    ldx   #0
cset28 lda
      bscren+$800,x
    sta   attr1,x
    lda
      bscren+$900,x
    sta
      attr1+$100,x
    lda
      bscren+$a00,x
    sta
      attr1+$200,x
    lda
      bascren+$b00,x
    sta
      attr1+$300,x
    inx
    bne   cset28
cset30 lda
      %00000001
    ora   #$01
    sta   $01
zbasic jmp
      xbasic
```

Module 4

```
*=cpixad
scrnad=frekzp+2
zxchar=frekzp
zychar=frekzp+1
    lda   #multic
    bit   vicctx
    beq   phires
    asl   xplot
    rol   xhigh
ypixpo=scrnad
phires lda xplot
    and   #%111
    eor   #%111
    sta   usebit
    lda   yplot
    and   #%111
    asl   a
    asl   a
    sta   ypixpo
plot10 lda xhigh
    lsr   a
    lda   xplot
```

```
       ror   a
       lsr   a
       lsr   a
       sta   zxchar
       lda   yplot
       lsr   a
       lsr   a
       lsr   a
       sta   zychar
       lsr   a
       ror   scrnad
       lsr   a
       ror   scrnad
       clc
       adc   zychar
       adc   #>hibase
       sta   scrnad+1
       lda   zxchar
       asl   a
       asl   a
       asl   a
       bcc   plot18
       inc   scrnad+1
       clc
plot18 adc
         scrnad
       sta   scrnad
       bcc   plot2Ø
       inc   scrnad+1
plot2Ø rts
*=getxy1
       jsr   chrget
       jmp   gexyØ1
*=getxy
       jsr   inposn
gexyØ1 jsr
         infacc
       jsr   facfix
       lda   fixhi
       bmi   perror
       cmp   #>xmax
       bcc   gexyØ2
         bne   perror
       lda   fixlow
       cmp   #<xmax
       bcs   perror
gexyØ2 jsr
         inposn
       jsr   infix
       cpx   xymax+1
       bcs   perror
       rts
*=gptype
       jsr   inposn
       jsr   infix
       cpx   #3
       bcc   gptyØ2
       lda   #multic
       bit   vicctx
       beq   perror
       cpx   #5
       bcc   gptyØ2
perror pla
       pla
       ldx   orange
       jmp   (ierror)
gptyØ2 stx ptype
       rts
*=plot
       lda   #basic
       sta   exitpl
       jsr   getxy1
       lda fixlow
       sta   xplot
       lda   fixhi
       sta   xhigh
       stx   yplot
       jsr   gptype
       jmp   plotex
*=plotex
       jsr   cpixad
       lda   yplot
       cmp   #ymax+1
       bcc   plotØ4
       jmp   plot78
plotØ4 lda xhigh
       bmi   plotØ6
       cmp   #>max
       bcc   plotØ8
       bne   plotØ6
       lda   xplot
       cmp   #<xmax
       bcc   plotØ8
plotØ6 jmp
         plot78
plotØ8 lda
         #multic
       bit   vicctx
       bne   pmulti
       lda   ptype
       cmp   #Ø
       bne   plot24
       ldy   #Ø
       ldx   usebit
       lda   (scrnad),y
       and   offmsk,x
       sta   (scrnad),y
       jmp   plot5Ø
plot24 cmp #2
       bne   plot26
p2hres ldy #Ø
       ldx   usebit
       lda   (scrnad),y
       eor   bitmsk,x
       sta   (scrnad),y
       jmp   plot5Ø
plot26 ldy #Ø
       ldx   usebit
       lda   (scrnad),y
       ora   bitmsk,x
       sta   (scrnad),y
       jmp   plot5Ø
pmulti lda
         usebit
       lsr   a
       sta   usebit
       lda   ptype
       cmp   #Ø
       bne   plot32
       jsr   pØmlti
       jmp   plot5Ø
plot32 cmp #4
       bne   plot34
       ldy   #Ø
       ldx   usebit
       lda   (scrnad),y
       eor   allmul,x
       sta   (scrnad),y
       jmp plot5Ø
plot34 cmp #1
       bne   plot36
       jsr   pØmlti
       ora   onmul1,x
       sta   (scrnad),y
       jmp   plot5Ø
plot36 cmp #2
       bne   plot38
       jsr   pØmlti
       ora   onmul2,x
       sta   (scrnad),y
       jmp   plot5Ø
plot38 jsr
         pØmlti
       ora   onmul3,x
       sta   (scrnad),y
plot5Ø lda
         scrnad+1
       eor   #>hibase
       sta   scrnad+1
       lsr   scrnad+1
       ror   scrnad
       lsr   scrnad+1
       ror   scrnad
       lsr   scrnad+1
       ror   scrnad
       lda   #>attr1
       eor   scrnad+1
       sta   scrnad+1
       ldy   #Ø
       lda   inkx12
       cmp   #$ff
       bne   plot64
       lda   #multic
       bit   vicctx
       beq   plot78
       lda   ink12
       sta   (scrnad),y
       clc
       lda   scrnad+1
adc
         #>cmemry->
       attr1
sta    scrnad+1
       lda   ink3
       sta   (scrnad),y
       bpl   plot78
plot64 nop
       lda   inkx12
       sta   (scrnad),y
       clc
       lda   scrnad+1
       adc
         #>memry->
         attr1
       sta   scrnad+1
       lda   inkx3
       sta   (scrnad),y
plot78 lda
         exitpl
       beq   plot8Ø
       rts
plot8Ø jmp
         xbasic
pØmlti ldy #Ø
       ldx   usebit
       lda   (scrnad),y
       and   offmul,x
       sta   (scrnad),y
       rts
*=bitmsk
  byt
     $Ø1,$Ø2,$Ø4,$Ø8,
     $1Ø,$2Ø,$4Ø,
     $8Ø
offmsk byt
     $fe,$fd,$fb,$f7,
     $ef,$df,$bf,
     $7f
offmul byt
     $fc,$f3,$cf,$3f
allmul byt
     $Ø3,$Øc,$3Ø,$cØ
onmul1 byt
     $Ø2,$Ø8,$2Ø,$8Ø
onmul2 byt
     $Ø1,$Ø4,$1Ø,$4Ø
onmul3 byt
     $Ø3,$Øc,$3Ø,$cØ
```

MODULE 5

```
xdiff = $Ø33c
xdiffh = $Ø33d
ydiff = $Ø33e
xdiag = $Ø33f
xdiagh = $Ø34b
ydiag = $Ø34Ø
xpara = $Ø341
xparah = $Ø34c
ypara = $Ø342
xdist = $Ø343
xdishi = $Ø344
ydist = $Ø345
ydishi = $Ø346
*=line
       lda   #linejb
       sta   exitpl
       jsr   getxy1
       lda   fixlow
       sta   xplot
       lda   fixhi
       sta   xhigh
       stx   yplot
       jsr   getxy
       lda   fixlow
       sta   xthere
       lda   fixhi
       sta   xthrhi
       stx   ythere
       jsr   gptype
       lda   xthere
       sec
       sbc   xplot
       sta   xdiff
       lda   xthrhi
       sbc   xhigh
       sta   xdiffh
       bcs   lineØ6
       lda   #minus1
       sta   xdiag
       sta   xdiagh
       jmp   lineØ8
lineØ6 lda
         #<plus1
       sta   xdiag
       lda   #>plus1
       sta   xdiagh
lineØ8 lda
         ythere
       sec
       sbc   yplot
       sta   ydiff
       bcs   line1Ø
       lda   #minus1
       sta   ydiag
       jmp   line12
line1Ø lda
         #plus1
       sta   ydiag
line12 lda xdiag
       sta   xpara
       lda   xdiagh
       sta   xparah
       lda   #Ø
       sta   ypara
       lda   xdiag
       cmp   #minus1
       beq   line14
       lda   xdiff
       sta   xdist
       lda   xdiffh
       sta   xdishi
       jmp   line16
line14 lda xplot
       sec
       sbc   xthere
       sta   xdist
       lda   xhigh
       sbc   xthrhi
       sta   xdishi
line16 lda ydiag
       cmp   #minus1
       beq   line18
       lda   ydiff
       sta   ydist
       lda   #Ø
       sta   ydishi
       jmp   line2Ø
line18 lda yplot
       sec
       sbs   ythere
       sta   ydist
       lda   #Ø
       sta   ydishi
line2Ø lda xdist
       sec
       sbc   ydist
       lda   xdishi
       sbc   ydishi
       bcs   line22
       lda   #Ø
       sta   xpara
       sta   xparah
       lda   ydiag
       sta   ypara
       lda   xdist
       ldx   ydist
       sta   ydist
       stx   xdist
       lda   xdishi
       ldx   ydishi
       stx   xdishi
       sta   ydishi
bigdis = $Ø343
bigdhi = $Ø344
tinyd = $Ø345
tinyhi = $Ø346
dodiag = $Ø347
dodihi = $Ø348
stepct = $Ø349
stephi = $Ø34a
xnow = $cfdØ
xnowhi = $cfd1
ynow = $cfd2
line22 lda
         bigdhi
       ror   a
       lda   bigdis
       ror   a
       sta   dodiag
       lda   #Ø
       sta   stepct
       sta   stephi
       sta   dodihi
       lda   xplot
```

```
 sta  xnow
 lda  xhigh
 sta  xnowhi
 lda  yplot
 sta  ynow
line26 lda xnow
 sta  xplot
 lda  xnowhi
 sta  xhigh
 lda  ynow
 sta  yplot
 jsr  plotex
 lda  dodiag
 clc
 adc  tinyd
 sta  dodiag
 lda  dodihi
 adc  tinyhi
 sta  dodihi
 lda  dodiag
 sec
 abc  bigdis
 lda  dodihi
 sbc  bigdhi
 bcc  line28
 lda  dodiag
 sec
 sbc  bigdis
 sta  dodiag
 lda  dodihi
 sbc  bigdhi
 sta  dodihi
 lda  xnow
 clc
 adc  xdiag
 sta  xnow
 lda  xnowhi
 adc  xdiagh
 sta  xnowhi
 lda  ynow
 clc
 adc  ydiag
 sta  ynow
 jmp  line30
line28 lda xnow
 clc
 adc  xpara
 sta  xnow
 lda  xnowhi
 adc  xparah
 sta  xnowhi
 lda  ynow
 clc
 adc  ypara
 sta  ynow
line30 inc
   stepct
 bne  line31
 inc  stephi
line31 lda
   stephi
 cmp  bigdhi
 bcs  line32
 jmp  line26
line32 lda
   stepct
 cmp  bigdis
 bcs  line34
 jmp  line26
line34 lda
   exitpl
 cmp  #linejb
 bne  line36
 jmp  xbasic
line36 rts
```

MODULE 6

```
*=block
 lda  #rtsjob
 sta  exitpl
 jsr  getxy1
 lda  fixlow
 sta  xlftlo
 lda  fixhi
 sta  xlfthi
 stx  ytop
 jsr  getxy
 lda  fixlow
 sta  xrgtlo
 lda  fixhi
 sta  xrgthi
 stx  ybottm
 jsr  gptype
 lda  ybottm
 sec
 sbc  ytop
 beq  blok99
 bcc  blok99
 lda  ytop
 sta  ynow
block20 lda
   xlftlo
 sta  xplot
 lda  xlfthi
 sta  xhigh
 lda  xrgtlo
 sta  xthere
 lda  xrgthi
 sta  xthrhi
 lda  ynow
 sta  ythere
 sta  yplot
 cmp  ybottm
 beq  blok99
 inc  ynow
 jsr  linex
 jmp  blok20
blok99 jmp
   xbasic
xlftlo byt Ø
xlfthi byt Ø
ytop byt Ø
xrgtlo byt Ø
xrgthi byt Ø
ybottm byt Ø
ynow byt Ø
```

MODULE 7

```
*=char
csize=8
 lda  #linejb
 sta  exitpl
 jsr  getxy1
 lda  fixlow
 sta  xchrlo
 lda  fixhi
 sta  xcharh
 stx  ychar
 jsr  inposn
 jsr  infix
 stx  pokeco
 jsr  gptype
 jsr  cget
 txa
 beq  charØ8
 sta  height
 jsr  char1Ø
charØ8 jmp
   xbasic
char1Ø lda
   pokeco
 asl  a
 asl  a
 asl  a
 sta  char2Ø+1
 lda  pokeco
 lsr  a
 lsr  a
 lsr  a
 lsr  a
 lsr  a
char11 ora
   #>ch1bse
 sta  char2Ø+2
char14 ldx #Ø
   start at byte Ø
 lda  ctimer
 and
   #%11111111-
   timbit
 sta  ctimer
 lda  $Ø1
 and  #%11111Ø11
 sta  $Ø1
char2Ø lda
   $ffff,x
 sta  pattrn,x
 inx
 cpx  #csize
 bne  char2Ø
 lda  $Ø1
 ora  #%ØØØØØ1ØØ
 sta  $Ø1
 inc  ctimer
 ldx  #Ø
char22 stx
   rowcnt
 lda  pattrn,x
 jsr  char3Ø
 ldx  rowcnt
 inx
 cpx  #csize
 bne  char22
char24 rts
char3Ø sta
   rpatrn
 ldx  height
 stx  thisct
char32 lda
   xchrlo
 sta  xnowlo
 sta  xplot
 lda  xcharh
 sta  xnowhi
 sta  xhigh
 ldy  #csize
char34 sty
   bitcnt
 lda  ychar
 sta  yplot
 lda  rpatrn
 and  bitmsk-1,y
 beq  char36
 jsr  plotex
char36 lda
   xnowlo
 clc
 adc  #1
 sta  xnowlo
 sta  xplot
 lda  xnowhi
 adc  #Ø
 sta  xnowhi
 sta  xhigh
 ldy  bitcnt
 dey
 bne  char34
 inc  ychar
 dec  thisct
 bne  char32
char39 rts
thisct byt Ø
rpatrn byt Ø
rowcnt byt Ø
bitcnt byt Ø
xchrlo byt Ø
xcharh byt Ø
xnowlo byt Ø
xnowhi byt Ø
ychar byt Ø
height byt Ø
pokeco byt Ø
pattrn = $Ø2a7
   !89 byte work
   area
ctrla = $Ø1
ctrlb = $Ø2
rvson = $12
rvsoff= $92
*=text
 lda  #linejb
 sta  exitpl
 jsr  getxy1
 lda  fixlow
 sta  xchrlo
 lda  fixhi
 sta  xcharh
 stx  ytext
 lda  #vstrng
 sta  valtyp
 sta  intflg
 lda  #<argho
 ldx  #>argho
 sta  forpnt
 stx  forpnt+1
 stx  txtptr
 jsr  inposn
 jsr  instrg
 lda  argho
 sta  txtlen
 lda  argho+1
 sta  text12+1
 lda  argho+2
 sta  text12+2
 jsr  gptype
   jsr  cget
 txa
 beq  textØ9
 sta  height
 jsr  cget
 stx  txtwid
 jmp  text1Ø
textØ9 jsr infix
 jmp  xbasic
text1Ø ldx
   txtlen
 bne  text11
 jmp  text95
text11 ldx
   txtptr
text12 lda
   $ffff,x
 sta  txbyte
 cmp  #rvson
 beq  text14
 cmp  #rvsoff
 beq  text16
 cmp  #'+Ø
 bcs  text2Ø
 cmp  #ctrla
 beq  text17
 cmp  #ctrlb
 beq  text18
 jmp  text9Ø
text14 lda
   #%1ØØØØØØØ
 sta  revflg
 jmp  text9Ø
text16 lda #Ø
 sta  revflg
 jmp  text9Ø
text17 lda
   #>ch1bse
 sta  char11+1
 jmp  text9Ø
text18 lda
   #>ch2bse
 sta  char11+1
 jmp  text9Ø
text2Ø and
   #%1ØØØØØØØ
 lsr  a
 sta  shift
 lda  txbyte
 and  #%ØØ111111
 ora  shift
 ora  revflg
 sta  pokeco
 stx  txtptr
 lda  ytext
 sta  ychar
 jsr  char1Ø
 lda  xchrlo
 clc
 adc  txtwid
 sta  xcharlo
 lda  xcharh
 adc  #Ø
 sta  xcharh
text9Ø ldx
   txtptr
 inx
 cpx  txtlen
 beq  text92
 stx  txtptr
 jmp  text11
text92 lda
   #>ch1bse
 sta  char11+1
text95 jmp
   xbasic
ytext byt Ø
txtwid byt Ø
revflg byt Ø
txtptr byt Ø
txtlen byt Ø
txbyte byt Ø
shift byt Ø
```

MODULE 8

```
*=draw
drxlow=holder
drxhi=holder+1
drawy=holder+2
drcode=holder+3
bytsdn=holder+4
drawln=holder+5
todo=holder+6
ypart=holder+7
xpart=holder+8
    jsr  chrget
    lda  #linejb
    sta  exitpl
    lda  #vstrng
    sta  valtyp
    sta  intflg
    lda  #<argho
    ldx  #>argho
    sta  forpnt
    stx  forpnt+1
    stx  bytsdn
    jsr  instrg
    lda  argho
    sta  drawln
    lda  argho+1
    sta  draw22+1
    lda  argho+2
    sta  draw22+2
    jsr  getxy
    lda  fixlow
    sta  drxlow
    lda  fixhi
    sta  drxhi
    stx  drawy
    jsr  gptype
    ldy  #$8Ø
    sty  ypart
    sty  xpart
draw2Ø ldy
       drsize
    sty  todo
bitop=$2c
draw21 lda
       #bitop
    sta  draw61
    ldx  bytsdn
    cpx  drawln
    bpl  draw49
draw22 lda
       $ffff,x
    sta  drcode
    cmp  #'9
    beq  draw49
    cmp  #'Ø
    beq  draw5Ø
    cmp  #'1
    beq  draw51
    cmp  #'2
    beq  draw27
    cmp  #'3
    beq  draw28
    cmp  #'5
    beq  draw5Ø
    cmp  #'6
    beq  draw51
    cmp  #'7
draw27 beq
       draw52
    cmp  #'8
draw28 bne
       draw48
    jmp draw53
draw48 jmp $cee1
draw49 jmp
       xbasic
x1pixl = tbxpix
x2pixl =
       tbxpix+2
x3pixl =
       tbxpix+4
xØpixl =
       tbxpix+6
ylpixl = tbypix
y2pixl =
       tbypix+1
y3pixl =
       tbypix+2
yØpixl =
       tbypix+3
x3part = tbxpar
xØpart =
       tbxpar+1
x1part =
       tbxpar+2
x2part =
       tbxpar+3
y3part = tbypar
yØpart =
       tbypar+1
y1part =
       tbypar+2
y2part =
       tbypar+3
draw5Ø lda xpart
    clc
    adc  xØpart
    sta  xpart
    lda  drxlow
    adc  xØpixl
    jsr  switox
    adc  xØpixl+1
    sta  drxhi
    lda  ypart
    clc
    adc  yØpart
    sta  ypart
    lda  drawy
    adc  yØpixl
    jmp  draw59
draw51 lda xpart
    clc
    adc  x1part
    sta  xpart
    lda  drxlow
    adc  x1pixl
    jsr  switox
    adc  x1pixl+1
    sta  drxhi
    lda  ypart
    clc
    adc  y1part
    sta  ypart
    lda  drawy
    adc  y1pixl
    jmp  draw59
draw52 lda xpart
    clc
    adc  x2part
    sta  xpart
    lda  drxlow
    adc  x2pixl
    jsr  switox
    adc  x2pixl+1
    sta  drxhi
    lda  ypart
    clc
    adc  y2part
    sta  ypart
    lda  drawy
    adc  y2pixl
    jmp  draw59
draw53 lda xpart
    clc
    adc  x3part
    sta  xpart
    lda  drxlow
    adc  x3pixl
    jsr  switox
    adc  x3pixl+1
    sta  drxhi
    lda  ypart
    clc
    adc  y3part
    sta  ypart
    lda  drawy
    adc  y3pixl
draw59 cmp drawy
    beq  draw6Ø
    sta  drawy
    lda  #jsrop
    sta  draw61
draw6Ø lda
       drcode
    cmp  #'4
    bmi  draw62
    lda  drxlow
    sta  xplot
    lda  drxhi
    sta  xhigh
    lda  drawy
    sta  yplot
    stx  bytsdn
draw61 jsr plotex
draw62 dec todo
    beq  draw64
    jmp  draw21
draw64 inc
       bytsdn
    jmp  draw2Ø
jsrop=$2Ø
switox php
    cmp  drxlow
    beq  swix99
    pha
    lda  #jsrop
    sta  draw61
    pla
    sta  drxlow
swix99 lda drxhi
    plp
    rts
    end
```

MODULE 9

```
*=rec
    lda  #rtsjob
    sta  exitpl
    jsr  getxy1
    lda  fixlow
    sta  xleft
    lda  fixhi
    sta  xlfthi
    stx  ytop
    jsr  getxy
    lda  fixlow
    sta  width
    lda  fixhi
    sta  widhi
    stx  rhight
    jsr  gptype
    lda  xleft
    clc
    adc  width
    sta  xright
    lda  xlfthi
    adc  widhi
    sta  xrthi
    cmp  #>xmax
    bmi  rectØ4
    bne  rectØ9
    lda  xright
    cmp  #<xmax
    bpl  rectØ9
rectØ4 lda ytop
    clc
    adc  rhight
    sta  ybottm
    bcs  rectØ9
    cmp  #ymax+1
    bcc  rect1Ø
rectØ9 ldx
       #serror
    jmp  (ierror)
rect1Ø lda xleft
    sta  xplot
    lda  xlfthi
    sta  xhigh
    lda  ytop
    sta  yplot
    sta  ythere
    lda  xright
    sta  xthere
    lda  xrthi
    sta  xthrhi
    jsr  lineex
    lda  xright
    sta  xplot
    sta  xthere
    lda  xrthi
    sta  xhigh
    sta  xthrthi
    lda  ybottm
    sta  ythere
    lda  ytop
    sta  yplot
    jsr  lineex
    lda  ybottm
    sta  yplot
    sta  ythere
    lda  xleft
    sta  xthere
    lda  xlfthi
    sta  xthrhi
    lda  xright
    sta  xplot
    lda  xrthi
    sta  xhigh
    jsr  lineex
    lda  ytop
    sta  ythere
    lda  ybottm
    sta  yplot
    lda  xleft
    sta  xplot
    sta  xthere
    lda  xlfthi
    sta  xhigh
    sta  xthrhi
    jsr  lineex
    jmp  xbasic
xleft=holder
xlfthi=holder+1
ytop=holder+2
width=holder+3
widhi=holder+4
rhight=holder+5
xright=holder+6
xrthi=holder+7
ybottm=holder+8
```

MODULE 10

```
topbsc=hibase-1
prtstr=$ab1e
prtnum=$bdcd
bytefr=$e46Ø
initio=$e518
chrout=$ffd2
fretop=$33
memsiz=$37
vartab=$2d
blue=6
grey3=15
bluprt=31
rvson=18
*=welcom
    lda  #<topbsc
    sta  fretop
    sta  memsiz
    lda  #>topbsc
    sta  fretop+1
    sta  memsiz+1
    jsr  initio
    lda  #blue
    sta  border
    lda  #grey3
    sta  bkgrnd
    ldx  #Ø

initØ2 lda
       initØ4,x
    jsr  chrout
    inx
    cpx
       #initØ6-initØ4
    bne  initØ2
    lda  memsiz
    sec
    sbc  vartab
    tax
    lda  memsiz+1
    sbc  vartab+1
    jsr  prtnum
    lda  #<bytefr
    ldy  #>bytefr
    jsr  prtstr
    jmp  modul1
initØ4 byt
       bluprt
    byt  rvson
    txt  "input
         hires graphics"
    txt  "          "
initØ6=*
rotang=exitpl
angmax=7
*=rot
    jsr  infix1
    cpx  #angmax+1
    bpl  rotnØ9
    stx  rotang
    txa
    asl  a
    tax
```

```
    bpl   rotn1Ø
rotnØ9 lda
      #serror
    jmp   (ierror)
rotn1Ø lda
      xpixØ,x
    sta   tbxpix
    lda   xpixØ+1,x
    sta   tbxpix+1
    lda   xpix2,x
    sta   tbxpix+4
    lda   xpix2+1,x
    sta   tbxpix+5
    lda   xpix3,x
    sta   tbxpix+6
    lda   xpix3+1,x
    sta   tbxpix+7
    lda   xpix1,x
    sta   tbxpix+2
    lda   xpix1+1,x
    sta   tbxpix+3
    ldx   rotang
    lda   ypixØ,x
    sta   tbypix
    lda   ypix1,x
    sta   tbypix+1
    lda   ypix2,x
    sta   tbypix+2
    lda   ypix3,x
    sta   tbypix+3
    lda   xparØ,x
    sta   tbxpar
    lda   xpar1,x
    sta   tbxpar+1
    lda   xpar2,x
    sta   tbxpar+2
    lda   xpar3,x
    sta   tbxpar+3
    lda   yparØ,x
    sta   tbypar
    lda   ypar1,x
    sta   tbypar+1
    lda   ypar2,x
    sta   tbypar+2
    lda   ypar3,x
    sta   tbypar+3
    jsr   inposn
    jsr   infix
    stx   drsize
    txa
    bne   rotn9Ø
    jmp   rotnØ9
rotn9Ø jmp
      xbasic
    p=1
m=%1111111111111
    111
n=%11111111
xpixØ wor Ø,Ø
xpix3 wor p,Ø
xpix1 wor Ø,m
xpix2 wor
      m,m,Ø,Ø,p,Ø,Ø,m
ypixØ byt n,n
ypix3 byt Ø,Ø
ypix1 byt p,Ø
ypix2 byt
      Ø,n,n,n,Ø,Ø,p,Ø
r=$b6
s=$4b
xparØ = *
yPar2 byt Ø,s
xpar2 = *
ypar1 byt Ø,r
xpar1 = *
ypar3 byt Ø,r
xpar3 = *
yparØ byt
      Ø,s,Ø,s,Ø,r,Ø,r
```

MODULE 11

```
result=somwer
*=test
    jsr   getxy1
    lda   fixlow
    sta   xplot
    lda   fixhi
    sta   xhigh
    stx   yplot
    jsr   ctestp
    jmp   xbasic
*=ctestp
    jsr   cpixad
    ldy   #Ø
    lda   vicctx
    and   #multic
    bne   test2Ø
    ldx   usebit
    lda   bitmsk,x
test1Ø and
      (frekzp+2),y
    beq   test14
test12 lda
      #plus1
test14 sta
      result
    rts
test2Ø lda
      usebit
    lsr   a
    tax
    ldy   #Ø
    lda   bitmsk+32,x
    bne   test1Ø
inttim=$ea31
loresp=frekzp
colorp=frekzp+2
lorend=attr4+qua
      rtr
flashØ lda
      jiffes
    clc
    adc   #1
    adc   #Ø
    sta   jiffes
    cmp   intrvl
    bne   flash9
    lda   #Ø
    sta   jiffes
    lda   frekzp
    pha
    lda   frekzp+1
    pha
    lda   frekzp+2
    pha
    lda   frekzp+3
    pha
    tya
    pha
    lda
      #%1111111+icle
      ar
    sta   ciamsk
    lda   #Ø
    sta   loresp
    sta   colorp
    lda   #>lobase
    sta   loresp+1
    lda   #>cmemry
    sta   colorp+1
    ldy   #Ø
flash3 lda
      (colorp),y
    and   #nybble
    cmp   colorf
    bne   flash4
    lda   (loresp),y
    eor   #$8Ø
    sta   (loresp),y
flash4 iny
    bne   flash6
    inc   loresp+1
    inc   colorp+1
flash6 cpy
      #<lorend
    bne   flash3
    lda   loresp+1
    cmp   #>lorend
    bne   flash3
    pla
    tay
    pla
    sta   frekzp+3
    pla
    sta   frekzp+2
    pla
    lda   frekzp+1
    pla
    sta   frekzp
    lda
      #iset+timera
    sta   ciamsk
flash9 jmp
      inttim
colorf byt $ff
jiffes byt 1
intrvl byt Ø
cinv=$Ø314
*=flash
    jsr   cget1
    stx   colorf
    jsr   inposn
    jsr   infix
    stx   intrvl
    lda   #Ø
    sta   jiffes
    lda   cinv
    cmp   #<flashØ
    beq   flashx
    lda
      #%1111111+icle
      ar
    sta   ciamsk
    lda   #<flashØ
    sta   cinv
    lda   #>flashØ
    sta   cinv+1
    lda
      #iset+timera
    sta   ciamsk
flashx jmp
      xbasic
*=off
    jsr   chrget
    lda   #Ø
    sta   intrvl
    jmp   xbasic
    end
```

MODULE 12

```
*=circle
    jsr   parms1
    jsr   parmsc
arcØ4 jsr parms2
    beq   errorx
    jsr   circ1Ø
    jmp   xbasic
errorx jmp
      (ierror)
*=arc
    jsr   parms1
    jsr   parmsa
    jmp   arcØ4
*=angl
anglØ2 jsr
      parms1
    jsr   gangle
    lda   fixlow
    sta   angl1
    sta   angle
    lda   fixhi
    sta   anglhi
    jsr   parms2
    beq   errorx
anglØ4 lda
      xthere
    sta   xplot
    lda   xthrhi
    sta   xhigh
    lda   ythere
    sta   yplot
    jsr   plotex
bitop = $2c
    lda   #bitop
    jsr   circ2Ø
    jmp   lineex
parms1 lda
      #linejb
    sta   exitpl
    jsr   getxy1
    lda   fixlow
    sta   xthere
    lda   fixhi
    sta   xthrhi
    stx   ythere
    rts
parmsc lda #Ø
    sta   angl1
    sta   anglhi
    sta   angend+1
    sta   angehi+1
    lda   #plus1
    sta   incrmt+1
    rts
parmsa jsr
      gangle
    lda   fixlow
    sta   angl1
    lda   fixhi
    sta   anglhi
    jsr   gangle
    lda   fixlow
    sta   angend+1
    lda   fixhi
    sta   angehi+1
    jsr   inposn
    jsr   infix
    stx   incrmt+1
    rts
parms2 jsr getxy
    lda   fixlow
    sta   xrad
    lda   fixhi
    sta   xradhi
    stx   yrad
    jsr   gptype
valid lda xrad
    bne   valid8
    lda   yrad
valid8 rts
gangle jsr
      inposn
    jsr   infacc
    jsr   facfix
avalid lda fixhi
    cmp   #Ø
    beq   avalix
    lda   fixlow if
      sen>Ø,jun>=1Ø4
    cmp   #<36Ø
    bpl   avali2
avalix rts
avali2 lda
      fixlow
    sec
    sbc   #<36Ø
    sta   fixlow
    lda   fixhi
    sbc   #>36Ø
    sta   fixhi
    bne   avalid
    rts
*=circ1Ø
    lda   angl1
    sta   angle
jsrop = $2Ø
    lda   #jsrop
    jsr   circ2Ø
    jsr   circ31
    lda   #>sincos+1
    jsr   circ3Ø
    jsr   circ31
    lda   #>sincos+2
    jsr   circ3Ø
    jsr   circ31
    lda   angl1
    clc
incrmt adc #$ff
    sta   angl1
    bcc   circ14
    inc   anglhi
circ14 ldx
      anglhi
    beq   circ16
    cmp   #<36Ø
    bcc   circ16
    sbc   #<36Ø
    dec   anglhi
    sta   angl1
circ16 sec
angend sbc #$ff
    bcc   circ1Ø
    sbc   incrmt+1
    bcs   circ1Ø
    lda   anglhi
angehi cmp #$ff
    bne   circ1Ø
    rts
cospve=1
cosneg=Ø
sinpve=Ø
sinneg=1
circ2Ø sta
```

```
circ57
   lda   anglhi
   bne   circ26
   lda   angle
   cmp   #90
   bcs   circ22
   ldx   #sinpve
   ldy   #cospve
   bne   circ28
circ22 cmp #180
   bcs   circ24
   lda   #<270
   clc
   sbc   angle
   sta   angle
   ldx   #sinpve
   ldy   #cosneg
   beq   circ28
circ24 ldx
      #sinneg
   lda   angle
   sec
   sbc   #180
   sta   angle
   ldy   #cosneg
   beq   circ28
cir26 lda angle
   sec
   sbc   #<270
   bmi   circ24
   sta   angle
   lda   #180
   clc
   sbc   angle
   sta   angle
   ldx   #sinneg
   ldy   #cospve
circ28 stx
      sine+1
   sty   cosine+1
   lda   #>sincos
circ30 sta
      circ32+2
   sta   circ42+2
circ31 ldx angle
   lda   $01
   and   #%11111110
   sta   $01
circ32 lda
      $ff00,x
   sta   mpier+1
   lda   xrad
   sta   mcand+1
   lda   xradhi
   sta   mcanhi+1
   jsr   mulply
sine lda #$ff
   bne   circ35
   lda   prodlo
   clc
   adc   xthere
   tay
   lda   prodhi
   adc   xthrhi
   beq   circ39
   cmp   #>xmax+1
   bcs   circ44
   cpy   #<xmax
   bcc   circ39
   jmp   circ80
circ35 lda
      xthere
   sec
   sbc   prodlo
   tay
   lda   xthrhi
   sbc   prodhi
   bcc   circ80
circ39 sta xhigh
   sty   xplot
circ40 lda #180
   clc
   sbc   angle
   tax
circ42 lda
      $ff00,x
   sta   mpier+1
   lda   yrad
   sta   mcand+1
   lda   #0
   sta   mcanhi+1
   jsr   mulply
cosine lda #$ff
   bne   circ45
   lda   prodlo
   clc
   adc   ythere
   bcs   circ80
   cmp   #ymax
circ44 bcs
      circ80
   bcc   circ49
circ45 lda
      ythere
   sec
   sbc   prodlo
   bcc   circ80
circ49 sta yplot
   pha
   ldy   xplot
   sty   xcoord+1
   ldx   #5
circ54 pla
   pha
   cmp   ytabl,x
   bne   circ56
   tya
   cmp   xtabl,x
   beq   pindex
circ56 dex
   bpl   circ54
circ57 jsr plotex
pindex ldy #$05
   pla
   sta   ytabl,y
xcoord lda #$ff
   sta   xtabl,y
   dey
   bpl   circ58
   ldy   #5
circ58 sty
      pindex+1
circ80 lda angle
   sec
   sbc   #90
   bcs   circ82
   adc   #180
circ82 sta angle
   lda   $01
   ora   #%00000001
   sta   $01
   rts
mulply lda #0
   sta   prodlo
   sta   prodhi
mpier ldx
      #$ff
   beq   mulp99
mcanhi ldy #%1
   beq   mcan02
   stx   prodlo
mcan02 clc
mcand adc #$ff
   bcc   mcan04
   clc
   inc   prodlo
   bne   mcan04
   inc   prodhi
mcan04 dex
   bne   mcand
   and   #%10000000
   beq   mulp99
   inc   prodlo
   bne   mulp99
   inc   prodhi
mulp99 rts
xrad byt 0
xradhi byt 0
yrad byt 0
angle byt 0
prodlo byt 0
prodhi byt 0
angl1 byt 0
anglhi byt 0
ytabl byt
      0,0,0,0,0,0
xtabl byt
      0,0,0,0,0,0
   end
```

MODULE 13

```
dummy=minus1
*=paint
   lda   #linejb
   sta   exitpl
   ldx   #1
   stx   stack
   ldx   #dummy
   stx   pstack+$200
   jsr   getxy1
   lda   fixlow
   sta   xpin
   lda   fixhi
   sta   xpinhi
   stx   ypin
   jsr   gptype
   jmp   pain20
*=pain20
   lda   $01
   and   #%11111110
   sta   $01
pain22 inc ypin
   jsr   probe
   dec   ypin
   lda   xpin
   bne   pain25
   dec   xpinhi
pain25 dec xpin
   jsr   probe
   inc   xpin
   bne   pain27
   inc   xpinhi
pain27 dec ypin
   jsr   probe
   inc   ypin
   inc   xpin
   bne   pain29
   inc   xpinhi
pain29 jsr probe
   lda   xpin
   bne   pain31
   dec   xpinhi
pain31 dec xpin
   lda   xpinhi
   sta   xhigh
   lda   xpin
   sta   xplot
   lda   ypin
   sta   yplot
   jsr   plotex
   jsr   soff
   lda   ypin
   cmp   #dummy
   bne   pain22
pain99 lda $01
   ora   #%00000001
   sta   $01
   jmp   xbasic
soff ldx stack
   dex
   cpx   #0
   bne   off2
   lda   oflag
   cmp   #1
   bne   off2
   lda   #0
   sta   oflag
   dex
off2 lda
      pstack,x
   sta   xpin
   lda
      pstack+$100,x
   sta   xpinhi
   lda
      pstack+$200,x
   sta   ypin
   stx   stack
   rts
probe lda
      #multic
   bit   vicctx
   beq   probe2
   lda   #0
   sta   xhigh
   lda   xpin
   sta   xplot
   cmp   #xcmax
   bcc   probes
   rts
probe2 lda
      xpinhi
   sta   xhigh
   beq   probe4
   lda   xpin
   sta   xplot
   cmp   #<xmax
   bcc   probe6
   rts
probe4 lda xpin
   sta   xplot
probe6 lda ypin
   sta   yplot
   cmp   #ymax
   bcc   probe8
   rts
probe8 jsr
      ctestp
   lda   somwer
   beq   onto
   rts
onto lda #8
   sta   scount
   ldx   stack
   lda   xpin
   pha
   ldy   ypin
   iny
onto1 dex
   beq   onto9
   dec   scount
   beq   onto9
   tya
   sec
   sbc
      pstack+$200,x
   beq   xspot
   cmp   #2
   bcc   xnear
   bne   onto1
xspot pla
   pha
   cmp   pstack,x
   bne   onto1
onto5 lda xpinhi
   sbc
      pstack+$100,x
   bne   onto1
   beq   onto20
xnear pla
   pha
   sbc   pstack,x
   beq   onto5
   bcs   onto6
   cmp   #%11111110
   bcc   onto1
   bcs   onto5
onto6   cmp
      #%11111110
   bne   onto1
   lda
      pstack+$100,x
   bne   onto20
   beq   onto1
onto9 ldx stack
   inx
   bne   onto12
   inx
   stx   oflag
onto12 stx stack
   dex
onto20 pla
   sta   pstack,x
   lda   xpinhi
   sta
      pstack+$100,x
   dey
   tya
   sta
      pstack+$200,x
   rts
xpin byt 0
xpinhi byt 0
ypin byt 0
stack byt 0
oflag byt 0
scount byt 0
   end
```

DISK-EDITING UTILITIES

For disk drive users, here is a utility that gives you direct access to stored information, enabling you to amend it, or retrieve it in case of accidents.



- DIRECT ACCESS
- READING AND WRITING
- DISK LAYOUT
- THE DIRECTORY
- DISK FORMAT
- USING THE PROGRAM
- READING A SECTOR
- EDITING THE DISK
- REINSTATING A DELETED FILE
- REWRITING A CORRUPTED DISK



The business of writing and reading files on a disk is handled by the *disk operating system* (DOS) or *disk filing system* (DFS) used by your computer. Normally you simply do not have to concern yourself with the organization of information on a disk or the transfer of data to and from it—the DOS/DFS takes care of everything. The disks and the disk unit simply become an extension to the computer—a data storage device.

But DOSs/DFSs do offer you the possibility of accessing individual parts of the disk so that information can be extracted, manipulated or amended beyond a parent program—a very useful capability which opens up a number of interesting and important possibilities.

You can use this form of *direct access* to change directory entries, file names, file data, file links, or to salvage information. The last of these is perhaps the most useful of all. Direct access enables you—amongst other things—to 'unscratch' files which have been deleted or scratched, CLOSE unclosed files, or re-establish *sector pointers* which have been corrupted, so restoring the correct 'chaining' of sectors which go to make up a particular file. There are several other uses which make a disk monitor program a useful utility to have.

Direct access of individual information 'blocks' on a disk can be likened to the use of a monitor to examine and alter selected parts of memory. And, in fact, the program that follows looks very much like such a monitor when in use. Unfortunately, it is not possible to access sectors on a Microdrive tape using a BASIC program, so there is no Spectrum disk monitor program.

The essence of the disk monitor's (or *disk editor's*) operation is a buffer which provides temporary storage for data read from a disk or written to it. While in the buffer, the information can be amended as desired and this new information used to overwrite old on the disk.

DISK GEOGRAPHY

Some knowledge of the layout of data on the disk is essential if specific tracks and sectors are to be located. This layout is normally referred to as a disk's *format* and is established by the formatting routines of the DOS/DFS used by your computer.

Hex notation is used extensively in all references dealing with direct access work, and that's why the general descriptions that follow do so as well. And note that decimal notation must not be used in connection with the use of DMON and other programs of this type. So you will find a good set of hex-decimal and hex-ASCII conversion tables useful.

HANDS ON

You need a map of the disk format before you can use the disk monitor and this is given under each machine's section. But first a couple of points. If you are working on a disk containing important information, make a backup first! Then any mistakes will not be catastrophic.

Secondly, it may sound obvious but you cannot recover information which is not there. If, for example, you want to recover a 'lost' or 'scratched' file you can do so only if the data hasn't already been overwritten. When a file is scratched, the storage space once used by that file is released for future use and could well have been corrupted if further file writing has taken place in the interim.

It may, however, be possible to salvage some data by directing the file pointers to those sections that have not already been overwritten.

The 1541 is the dedicated disk unit for the Commodore 64 (and Vic 20). It has its own on-board disk operating system and can thus be considered a fairly 'intelligent' device in its own right. While the disk monitor can be adapted for use on other CBM drives, the track and sector information which follows applies specifically to this unit.

1541 disks are divided into 35 *tracks*, each containing from between 17 and 21 *sectors* depending on the track's physical location. In all, a total of 683 sectors exist of which a maximum of 664 are available for use. Each sector has 256 bytes of storage space.

Track 18 is occupied by what is called a *directory*. This is normally accessed simply by typing LOAD"$",8 followed by LIST—this

displays the program, sequential and other types of file present on the disk. Up to 144 directory entries (hence files) may be contained on the one disk.

Track 18 is the most common one to access indirectly using a disk monitor, typically to correct the various directory corruptions or mistakes that can take place. A typical 'file saver' is unscratching files accidentally discarded, and re-establishing pointers to avoid corrupted areas of the disk which are unsalvageable.

But, of course, to find your way round you need a 'map' or *format* of how and where information is located on the disk. Let's look first at track 18, the directory.

First there's what's called the *block availability map* (BAM), the purpose of which is to indicate just what sector blocks are free for use. The BAM is updated after every disk access.

The general layout of the BAM and directory track is:

TRACK 18 ($12) SECTOR Ø

Byte	*Purpose*
$ØØ–$Ø1	Track and sector of the first block of the directory
$Ø2	Has value $41 (ASCII character A to indicate 1541 format)
$Ø3	Zero flag (not important)
$Ø4–$8F	Bit map of free blocks (marked 1) and allocated blocks (marked Ø)
$9Ø–$FF	DIRECTORY HEADER
Header format:	
$9Ø–$A1	Disk title (padded with shifted spaces – $AØ)
$A2–$A3	ID marker
$A4	Spacer (shifted space – $AØ)
$A5–$A6	Format type ($32,$41 – 2A in ASCII)
$A7–$AA	Spacers (shifted spaces – $AØ)
$AB–$FF	Unused ($ØØ) except for BLOCKS FREE legend

The actual directory starts in track 18 sector 1 and, depending on the number of files, may extend to other sectors as well. All of the remainder of track 18 is set aside for this task. A maximum of eight files can be detailed in each sector.

TRACK 18 ($12) SECTOR 1

Byte	*Purpose*
$ØØ–$Ø1	Track and sector of next directory block
$Ø2–$1F	DETAILS OF FIRST FILE:
File format:	
$Ø2	File type (see below)
$Ø3–$Ø4	Track and sector of first file data block
$Ø5–$14	Name of first file (padded with shifted spaces – $AØ)
$15–$16	Used for relative files only (T&S of first side-sector block)
$17	Relative file record length
$18–$1B	Not used
$1C–$1D	Used for T&S of new file when 'save and replace' instruction—@—used
$1E–$1F	Number of blocks in the first file (lo-byte, hi-byte)
$2Ø–$21	Spacer (shifted spaces—$AØ)
$22–$3F	DETAILS OF SECOND FILE (file format as for first file)

The second file details are then followed by another two-byte spacer, and so the format continues for eight files per block. If no further files follow in a particular block, the rest of it remains filled with zeros.

The two bytes immediately preceding the file name are the pointers to the first data block of that particular file. For the first file these can be found at $Ø3 and $Ø4. The values (in hex) give, in turn, the track and sector. So $11 $Ø1 would point to track 17 sector 1. The format of a typical data block or sector (of 256 bytes) starts simply enough with the pointer data for the next block of data in the file, again in track/sector order. The remaining bytes are filled with data. The last block used by the file starts with $ØØ and the next byte value indicates the number of bytes of that block which are used.

```
10 PRINT"[CLR][π]":POKE53280,Ø:POKE53281,
   Ø:GOSUB24ØØ
20 DIMA(255),S(35):HX$="Ø123456789
   ABCDEF":SE=1:TR=18
30 FORI=1TO17:S(I)=21:NEXT
40 FORI=18TO24:S(I)=19:NEXT
50 FORI=25TO3Ø:S(I)=18:NEXT
60 FORI=31TO35:S(I)=17:NEXT
70 PRINT:GOSUB1ØØØ:PRINT"□"
80 A$=LEFT$(T$,1)
90 IFA$="P"THENGOSUB12ØØ:GOTO7Ø
100 IFA$="X"THENPRINT"BASIC":END
110 IFA$="$"THENGOSUB15ØØ:GOTO7Ø
120 IFA$>="Ø"ANDA$<="9"THEN
   GOSUB16ØØ:GOTO7Ø
130 IFA$="D"THENGOSUB65Ø:GOTO7Ø
140 IFA$="S"THENGOSUB17ØØ:GOTO7Ø
150 IFA$="E"THENGOSUB18ØØ:GOTO7Ø
170 IFA$="R"THENGOSUB19ØØ:GOTO7Ø
180 IFA$="W"THENGOSUB21ØØ:GOTO7Ø
190 IFA$="C"THENGOSUB23ØØ:GOTO7Ø
200 IFA$="H"THENGOSUB24ØØ:GOTO7Ø
210 PRINT".?UC?":GOTO7Ø
650 OPEN15,8,15:OPEN8,8,8,"#":PRINT
   #15,"U1:"8;Ø;18;Ø:CLOSE15:CLOSE8
655 OPEN1,8,2,"$"
660 FORX=1TO141:GET#1,A$:NEXT
670 T$(Ø)="DELETED":T$(1)="SEQ":
   T$(2)="PROGRAM":T$(3)="USER":
   T$(4)="RELATIVE"
680 J=17:GOSUB94Ø
690 N$=B$
700 J=2
710 GOSUB94Ø
720 I$=B$
730 GET#1,A$
740 J=2
750 GOSUB94Ø
760 O$=B$
770 FORL=1TO88
780 GET#1,A$
790 NEXT
800 PRINT"DISK NAME:"N$:PRINT"□□□
   □□□□ID:"I$:PRINT"□□□□□□
   □OS:"O$"[■]"
810 PRINT"LENGTH","TYPE","NAME[■]"
820 FORP=1TO8
830 GET#1,T$,A$,A$
840 IFT$=""THENT$=CHR$(128)
850 J=15
860 GOSUB94Ø
870 N$=B$
880 GET#1,A$,A$,A$,A$,A$,A$,A$,A$,A$,
   L$,H$
890 L=ASC(L$+CHR$(Ø))+256*ASC
   (H$+CHR$(Ø)):IFL=ØTHEN93Ø
900 IFSTTHENCLOSE1:RETURN
910 PRINTL*256,T$(ASC(T$)-128),N$
920 IFP<8THENGET#1,A$,A$
930 NEXT:GOTO82Ø
940 B$=""
950 FORL=ØTOJ
960 GET#1,A$
970 IFA$<>CHR$(96)THENIFA$<>CHR$
   (16Ø)THENB$=B$+A$
980 NEXT:RETURN
1000 T$="":PRINT".";
1010 PRINT"[□][▮▮]";:GETA$:IFA$=""
   THEN1Ø1Ø
1020 IFA$=CHR$(13)THEN11ØØ
1030 IFA$=CHR$(2Ø)THEN111Ø
1040 IFLEN(T$)>1ØTHEN1Ø1Ø
1050 IFA$="□"ORA$="$"THEN1Ø9Ø
1060 IFA$<"Ø"THEN1Ø1Ø
1070 IFA$>"Z"THEN1Ø1Ø
1090 T$=T$+A$:PRINTA$;:GOTO1Ø1Ø
1100 IFT$<>""THENRETURN
1110 IFT$=""THEN1Ø1Ø
1120 T$=LEFT$(T$,LEN(T$)-1)
1130 PRINTA$;:GOTO1Ø1Ø
```

```
1200 REM PRINT ROUTINE
1210 X$=MID$(T$,3,2):GOSUB1300:S=X
1220 X$=MID$(T$,6,2):GOSUB1300:F=X
1230 FORI=STOFSTEP9
1240 X=I:GOSUB1400:PRINTH$“:”;:FOR
   T=ØTO8:IFI+T>255THENPRINT“***”:
   RETURN
1250 X=A(I+T):GOSUB1400:PRINTH$
   “□”;:NEXT
1260 FORT=ØTO8:A=A(I+T):IFA<32OR
   A>91THENA=32
1270 PRINTCHR$(A);:NEXT:PRINT:NEXT:
   RETURN
1300 A$=LEFT$(X$,1);B$=RIGHT$(X$,1):
   FORI=1TO16
1310 IFA$=MID$(HX$,I,1)THENH=
   (I-1)*16
1320 IFB$=MID$(HX$,I,1)THENL=
   (I-1)
1330 NEXT:X=H+L:RETURN
1400 H=INT(X/16):L=(X-H*16)
1410 H$=MID$(HX$,H+1,1)+MID$(HX$,
   L+1,1):RETURN
1500 IFLEN(T$)=5THEN1540
1505 IFLEN(T$)<>3THENPRINT“.?SX?”;:
   RETURN
1510 X$=RIGHT$(T$,2)
1520 GOSUB1300
1530 PRINT“.DEC”X:RETURN
1540 X$=RIGHT$(T$,2):GOSUB1300
1550 M=X:X$=MID$(T$,2,2):GOSUB1300:
   PRINT“.DEC”256*X+M:RETURN
1600 V=VAL(T$):IFV>65535ORV<ØTHEN
   PRINT“.??”;:RETURN
1610 M=INT(V/256)
1620 N=V-M*256
1630 X=M:GOSUB1400:A$=H$:X=N:
   GOSUB1400:A$=A$+H$
1640 PRINT“.HEX□”A$:RETURN
1700 PRINT“LAST TRACK:$”;:X=TR:GOSUB
   1400:PRINTH$
1710 PRINT“□□□□SECTOR:$”;:X=SE:
   GOSUB1400:PRINTH$
1720 RETURN
1800 X$=MID$(T$,3,2)
1810 GOSUB1300
1820 A=X:X$=MID$(T$,6,2):GOSUB1300:
   B=X
1830 A(A)=B:PRINT“OK”:RETURN
1900 IFLEN(T$)=1THENGOSUB2000:
   RETURN
1910 X$=MID$(T$,3,2):GOSUB1300:IFX<1
   ORX>35THENPRINT“.?IT?”;:RETURN
1920 A=X:X$=MID$(T$,6,2):GOSUB1300:
   IFX<ØORX>S(A)THENPRINT“.?IS?”;:
   RETURN
1930 TR=A:SE=X:GOSUB2000
1940 RETURN
2000 OPEN15,8,15
2010 OPEN8,8,8,“#”
2020 PRINT#15,“U1:”8;Ø;TR;SE
2030 PRINT#15,“B-P:”8;Ø
2040 FORI=ØTO255:GET#8,A$:IFST<>Ø
   ANDST<>64THENPRINT“.?DR?”:
   CLOSE8:CLOSE15:RETURN
2050 A(I)=ASC(A$+CHR$(Ø))
2060 NEXT
2070 CLOSE8:CLOSE15:PRINT“OK”:RETURN
2100 IFLEN(T$)=1THENGOSUB2200:
   RETURN
2110 X$=MID$(T$,3,2):GOSUB1300:IFX<1
   ORX>35THENPRINT“.?IT?”;:RETURN
2120 A=X:X$=MID$(T$,6,2):GOSUB1300:
   IFX<ØORX>S(A)THENPRINT“.?IS?”;:
   RETURN
2130 TR=A:SE=X:GOSUB2200
2140 RETURN
2200 OPEN15,8,15
2210 OPEN8,8,8,“#”
2220 PRINT#15,“B-P:”8;Ø
2230 FORI=ØTO255:PRINT#8,CHR$(A(I));:
   IFST<>ØANDST<>64THENPRINT
   “.?DW?”:GOTO2250
2240 NEXT:PRINT#15,“U2:”8;Ø;TR;SE
2250 CLOSE8:CLOSE15
2260 RETURN
2300 OPEN15,8,15
2310 INPUT#15,A,B$,C,D
2320 PRINT“ERROR NO.:”A
2330 PRINT“□□□□□TYPE:□”B$
2340 PRINT“□□@□TRACK:”C
2350 PRINT“□□□SECTOR:”D
2360 CLOSE15:RETURN
2400 PRINT“■P□XX□XX□□ - PRINT
   MEMORY
2410 PRINT“D□□□□□□□□ -
   DIRECTORY
2420 PRINT“R□XX□XX□□ - READ
   FROM DISK
2430 PRINT“W□XX□XX□□ - WRITE TO
   DISK
2440 PRINT“E□XX□XX□□ - EDIT
   MEMORY
2450 PRINT“S□□□□□□□□□ - LAST
   SECTOR/TRACK
2460 PRINT“$□□□□□□□□□ - HEX
   TO DECIMAL
```



```
2470 PRINT"(NUMBER) — DECIMAL TO HEX
2480 PRINT"C□□□□□□□□ —
   LAST ERROR
2490 PRINT"X□□□□□□□□ — EXIT
   TO BASIC
2500 PRINT"H□□□□□□□□ —
   PRINTS MENU
2510 PRINT"■IT□ — ILLEGAL TRACK
2520 PRINT"IS□ — ILLEGAL SECTOR
2530 PRINT"SX□ — SYNTAX ERROR
2540 PRINT"UC□ — UNKNOWN COMMAND
2550 PRINT"DR□ — DISK READ ERROR
2560 PRINT"DW□ — DISK WRITE ERROR
2570 RETURN
```

USING THE COMMODORE DMON

When you RUN the program you'll see a menu offering eleven commands, as well as a list of error messages with their explanations. Each command is accessed by the letter shown, and the double Xs indicate you must input a number as explained below.

For a comprehensive directory, press D—the lengths of the programs are shown in bytes, and both the deleted and the current file names are shown.

To read any part of the disk, press R then enter the number of the track and the sector, separated by spaces. Remember to use hex! Try R 12 Ø1 to read in the directory as stored on the disk. The data read in is stored in the disk buffer and can be displayed on the screen by pressing P, followed by the number of the start and end byte of the section you want to view. To see the entire contents use P ØØ FF. The byte numbers are shown in the left-hand column, the contents in hex are shown in the centre and the ASCII translation is shown at the right.

To change a byte, press E, followed by the number of the byte you wish to change and the new value. You'll see the new value appear in the correct place on the screen. To write this back to disk, press W followed by the destination track and sector.

The other commands available are S, which prints out the last track and sector accessed; $ followed by a hex number to convert it to decimal; a decimal number on its own to convert it to hex; C to print the code of the last error; H to redisplay the menu, and X to return to BASIC.

When a file is scratched, the file type marker in the directory is altered. This marker immediately precedes the file Track and Sector (T&S) pointers which in turn come just before the file name. A directory listing can show several types of file, DEL, SEQ, PRG, USR and REL.

The files are normally open—which means ready for (over)writing—or closed. A file is opened when 'scratched' to release its allocation of storage blocks. When closed, a file is 'active' in the sense that it has been stored. But it is possible to lock these files so that they cannot be scratched easily.

Using the disk monitor you can locate the file type byte easily enough and examine its status. The following file types and hex value designations exist:

File type

	Closed	*Open*	*Protected*
DELeted	$8Ø	$ØØ	—
SEQuential	$81	$Ø1	$C1
PRoGram	$82	$Ø2	$C2
USeR	$83	$Ø3	$C3
RELative	$84	$Ø4	$C4

Thus a program file which has been scratched displays $Ø2 in the file type byte position. If the sectors or blocks of that file have not been overwritten, the file data can be recovered by using the disk monitor to alter the value to $82 (the value it would have if the file was active).

But by altering the value to $C2 you can actually lock the file to prevent scratching. And using the appropriate values in the third column, you can protect other types of file. On a subsequent directory listing, the file type letters will have a < next to the abbreviation to denote locking. If you did want to remove the files you can do so by NEWing the entire disk, or using the editor to change the values to the appropriate scratch value ('open').

If part of a file has been corrupted—as it would be if you were attempting to recover a scratched file some of whose allocation of blocks had been overwritten by an active file—you will have to 'follow through' the various T&S pointers to gauge the extent of the damage.

Start at the directory ($12 $Ø1) and establish the T&S pointer to the first data track of the file you're trying to recover. Then use the disk monitor to examine that track. If this appears intact—that is, it doesn't contain random garbage—proceed to examine the next one in the chain. The T&S location of this is given in the first two bytes of the sector you're currently examining.

If you come across a sector in the chain that is corrupted, first see whether repair work is possible simply by overwriting one or two of the earlier pointers. By rewriting an earlier pointer the file effectively skips the corrupted sector. This will enable you to recall information using the parent program, tidy up the information so that the end of one sector matches the start of the next, and reSAVE the whole file on a new disk.

The most common types of disk filing system (DFS) used with the BBC computer are Acorn's own and the Watford system. Both can make use of the disk monitor program, however, since the Watford DFS includes an EDIT command the following description is intended for the Acorn DFS.

Disks for use with the BBC have either 40 or 80 tracks, each containing ten sectors per track so you have 400 or 800 sectors on each side. Each sector is composed of 256 bytes of storage space.

Catalogue information (obtained using *CAT) accesses track Ø, sectors Ø and 1 whose format is shown below. In addition to all the file names (a maximum of 31), you can access (and so adjust) the pointers—sector references—of the data which goes to make up any one of those files. By reading and adjusting these directly you can recover programs or data which may be present but inaccessible by conventional means.

Track Ø SECTOR Ø FORMAT

Byte	*Purpose*
&ØØ–&Ø7	First eight characters of disk title, padded with spaces
&Ø8–&ØE	First file name, padded with spaces (seven characters max.)
&ØF	Directory letter of first file
&1Ø–&16	Second file name, padded with spaces (seven characters max.)
&17	Directory letter of second file

The eight-byte name and directory blocks continue up to a maximum of 31 files.

TRACK Ø SECTOR 1 FORMAT

Byte	*Purpose*
&ØØ–&Ø3	Last four characters of disk title (padded)
&Ø4	Count of number of write operations made to the disk
&Ø5	Eight-byte block count (should equal 8 times number of active files)
&Ø6	Individual bit settings (see below)
$Ø7	Number of sector on disk (eight LSBs of 1Ø-bit number)
&Ø8–&ØF	FIRST FILE STORAGE MAP

File format:

&Ø8–&Ø9	Load address, LSB first. This would be zero for a data file or &19ØØ for a BASIC program

&ØA–&ØB Execution address, LSB first. Zero for a data file, or &8Ø23 for a BASIC I or &8ØIF for BASIC II.

&ØC–&ØD File length (bytes), LSB first

&ØE Individual bit settings (see below)

&ØF Start sector (eight LSBs of 1Ø-bit number)

&1Ø–&17 SECOND STORAGE MAP

File format as above

And so on up to 31 files.

The individual bit settings in &Ø6 and &ØE are as follows. Setting bits 5 and 4 of &Ø6 give start-up option (!BOOT) while bits 1 and Ø are the two MSBs (most significant bits) of a 19-bit number. The remaining eight bits are stored in &Ø7.

The significance of bit settings in location ØE are as follows: 7 and 6 are two MSBs of 18-bit execution address (LSBs in &ØA and &ØB). Bits 5 and 4 provide the two MSBs of the file length (LSB in &ØC and &ØD) if required. Likewise bits 3 and 2 look after two MSBs for a load address if required. Bits 1,Ø provide the two MSBs for a 10-bit file start sector (LSBs in &ØF).

The start sector information in byte &ØF and bits 1 and Ø of byte &ØE give the starting point of the file. For example, if &ØF is 38 and the MSBs are zero, the file is located at &38/1Ø = 5.6, that is, track 5 sector 6. If &ØF is 43 and bit Ø in &ØE is set, the file is located at &143/1Ø, or track 32, sector 3.

It is useful to note that the load address (&ØE &Ø9) is normally &ØØØØ for a data file, &19ØØ for BASIC (remember, LSB first!). The execution address is again &ØØØØ for a data file and usually &8Ø23 for BASIC.

The appearance of the file map for BASIC program of 12Ø67 bytes might take the form: ØØ 19 23 8Ø 23 2F CC Ø2. This information can also be obtained in a slightly different form using the command *INFO*.*.

```
10 MODE3
20 DIM block 12
30 DIM buffer 255
40 printer=0
50 DR%=0
60 PROCload(DR%,0,0,1)
70 REPEAT
80 PROCprint
90 VDU28,0,24,79,21
100 PROCcommand
110 UNTIL com$="END□"
120 END
130 DEFPROCload(DR%,TR%,SCT%,RNW)
140 X%=block MOD 256:Y%=block DIV 256
150 A%=&7F
160 block?0=DR%
170 block!1=buffer
180 block?5=3
190 block?6=&4B+8*RNW
200 block?7=TR%
210 block?8=SCT%
220 block?9=&21
230 CALL&FFF1
240 ENDPROC
250 DEFPROCprint
260 CLS:VDU26
270 IF printer=1 THEN VDU2 ELSE VDU3
280 PRINT"Track□";TR%;"□□Sector□";
    SCT%;"□□Drive□";DR%
290 PRINT
300 FOR I%=0 TO 1
310 PRINT"□□□□□□□bytes (in hex)
    □□□□□□□□□□□ascii□□□";
320 NEXT
330 PRINT
340 line=32:pos=7:buff=buffer
350 FOR lin=1 TO line
360 L=LEN(STR$((lin-1)*8))
370 FOR I=1 TO 4-L:VDU32:NEXT
380 PRINT;STR$((lin-1)*8);"□□";
390 FOR linpos=0 TO pos
400 cont=linpos?buff
410 IF cont<&10 THEN PRINT"0";
420 PRINT;~cont;"□";
430 NEXT
440 PRINT"□";
450 FOR linpos=0 TO pos
460 cont=linpos?buff
470 asc=(cont>31 AND cont<127)
480 IF asc THEN PRINTCHR$cont; ELSE
    PRINT".";
490 NEXT
500 IF lin MOD2=0 THEN PRINT
510 buff=buff+8
520 NEXT
530 PRINT
540 VDU3
550 ENDPROC
560 DEFPROCcommand
570 INPUT"$"com$:comlen=LEN(com$):
    com$=com$+LEFT$("□□□□",4-
    comlen)
580 command=(INSTR("□□□□DRV TRK
    SCT INS PRT WRT END
    SHOW",com$,0))DIV4+1
590 ON command GOTO 600,610,620,630,
    640,650,660,680,670
600 PROCcommand:ENDPROC
610 PROCdrive:ENDPROC
620 PROCtrack:ENDPROC
630 PROCsector:ENDPROC
640 PROCinsert:ENDPROC
650 PROCprinter:ENDPROC
660 PROCload(DR%,TR%,SCT%,0):ENDPROC
670 PROCload(DR%,TR%,SCT%,1):ENDPROC
680 ENDPROC
690 DEFPROCdrive
700 INPUT"Drive□",DR%
710 ENDPROC
720 DEFPROCprinter
730 IF printer=1 THEN printer=0 ELSE
    printer=1
740 ENDPROC
750 DEFPROCinsert
760 REPEAT:INPUT"Offset from start
    ?...."os:UNTIL os>=0 AND os<256
770 REPEAT:INPUT"New value ?...."val$:val=
    EVAL(val$):UNTIL val>=0 AND val<256
780 buffer?os=val
790 ENDPROC
800 DEFPROCtrack
810 REPEAT
820 INPUT"Track□□?□"TR%
830 UNTIL TR% >= 0 AND TR%<80
840 PROCload(DR%,TR%,SCT%,1)
850 ENDPROC
860 DEFPROCsector
870 REPEAT
880 INPUT"Sector□□?□"SCT%
890 UNTIL SCT%>=0 AND SCT%<10
900 PROCload(DR%,TR%,SCT%,1)
910 ENDPROC
```

USING THE BBC DMON

As soon as you RUN the program you'll see a display of the bytes in track Ø, sector Ø, drive Ø along with an ASCII equivalent of the hex. To view any other part of the disk enter DRV to select the drive, TRK for the track, and SCT for the sector. Try swapping between sectors Ø and 1, which shows the file names and then, in the same positions, the file storage maps.

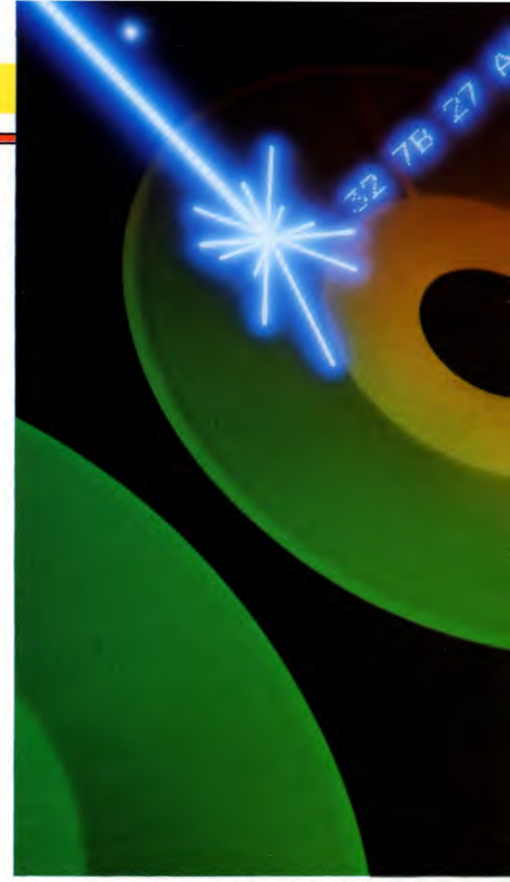
Type INS to insert (change) any of the bytes. First type in the number of the byte you wish to change, then the new value—in ASCII for the file names in sector Ø or in hex for the storage map in sector 1. Do not use decimal numbers.

When you've edited the sector you can write it back to the disk using WRT.

If you want a printout at any time type PRT to turn it on and PRT again to turn it off. If you change disks you may need to type SHOW to display the new disk, although the sector will normally be displayed automatically. Type END when you've finished.

The DMON program can be used for general maintenance work on disk files but perhaps the most important use is recovery of 'lost' files. First you need some information about the file length and, more important, the sector at which it begins. Under normal circumstances—that is, before a file is corrupted—this information can be obtained simply enough by keying *INFO <filename>.

Unless you already have this information logged (a good idea for any important file)



you are faced with a rather boring exploration of the disk to find, initially, the starting track and sector of the lost file and its length (&0C, &0D). Convert the track and sector information into a hexadecimal byte plus extra MSBs using the reverse of the procedure described on page 1613.

To recover the file, choose the next free 8-byte block in sector 0 and construct a suitable *new* filename plus directory letter (7+1 bytes), entered as ASCII codes of the characters. Be careful not to overwrite an existing active file's name.

Then use DMON to call up sector 1. Adjust &05 to cater for the extra file—simply increment it by 8.

You've created a new file and so a separate storage map—an extra eight bytes of data from &08 to &0F—is needed for the file. Put these into the first available 8-byte batch, corresponding to the file name. The first two bytes of this is the load address—&0000 (data) or &1900 (BASIC)—the next pair of bytes is the execution address.

Now use the editor to write in the length of the file on the next pair of bytes (LSB first!). If you don't know how long it is put in &00, &10 for the time being.

The one problem area is the setting of the value for the seventh byte, the two LSBs of which indicate the start sector of the file you are trying to recover. Remember, this information is already on the disk and a start point must be located and entered in the seventh byte before a file can be recovered.

When you've done this, exit the disk monitor. If you were unsure of the files length, first type PRINT~LOMEM and note down the figure. Then LOAD the program or data to check it has been recovered and find LOMEM again. The difference between these two values is the length of the file and can be inserted into bytes &0C and &0D of the storage map. Now *SAVE the file onto another disk just to make sure.

The Dragon Data disk drive has its own special interface containing the operating system, and the following information applies specifically to this unit.

The disks are divided into 40 or 80 tracks each containing 18 sectors of 256 bytes each.

The directory, which keeps track of all the files on the disk, is stored on tracks 16 and 20. These tracks are identical, but track 16 is used for the directory and track 20 for the system. The directory can be accessed by typing DIR, which then displays a list of all files with information on their type, length and number of bytes free.

Tracks 16 or 20 are the most common ones to access using the disk monitor, typically to reinstate an accidentally deleted file. But to find your way around you need a map of the information stored in the directory.

TRACK 16 SECTOR 3

Byte	*Purpose*
1	Descriptor code, eg 00 for valid file, 02 for protected file, 81 for deleted file
2–9	Name of file padded with zeros
10–12	File descriptor eg BAS, BAK
13–14	High and low bytes of 16 bit number giving the start track and sector
15	Total number of sectors used
16–24	Used to point to linked files
16,17	High and low bytes pointing to next start sector
18	Number of sectors used
19–21	Same for next section
22–24	Same for final section
25	Number of bytes used in last sector

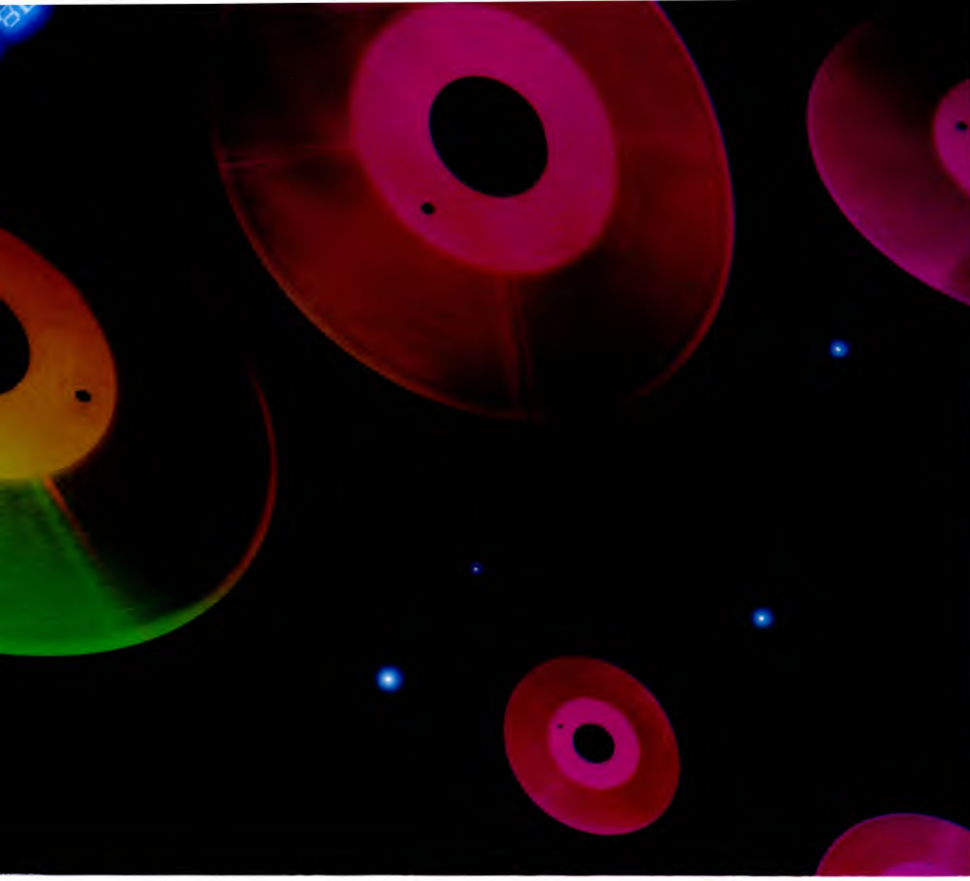
Sectors 1 and 2 of tracks 16 and 20 are used to tell the computer which sectors are in use. Each byte represents eight sectors, one bit per sector, counted from track zero, sector 1. Each bit starts off at 0 for a newly formatted disk but is set to 1 as that sector is filled. When a file is deleted the bit is changed back to zero.

The appearance of the directory entry for a BASIC file might take the form:

```
00 A B C 00 00 00 00 00 B A S 01 44 18 00 FC
09 00 00 00 00 00 00 DB
```

This is a file called 'ABC' starting at position &H0144,&H18 sectors long, with the second edition starting at &H00FC, 9 sectors long, and &HDB bytes in the last sector. The position is worked out as follows: &H0144=324 decimal, divide by 18 (there are 18 sectors per track) to find the track, equals 18. Since there's no remainder it starts at sector 1. The total length is &H18+&H8 sectors plus &HDB bytes, equals 8411 bytes.

```
10 CLEAR5000:DIMA$(1),D$(1),D(160):
   C$="↑"+CHR$(10)+CHR$(8)+
   CHR$(9)+"AH"+CHR$(13)+"□":D=1
20 CLS:PRINT@13,"menu"
30 PRINT@106,"lOAD SECTOR":PRINT
   @170,"vIEW/EDIT SECTOR":PRINT@234,
   "sAVE SECTOR":PRINT@298,"cATALOGUE"
40 R$=INKEY$:IFR$="" THEN40
50 R=INSTR("LVSC",R$):IFR=0 THEN40
60 IFSL=0AND(R=2ORR=3) THENPRINT:
   PRINT"NO SECTOR LOADED":FORK=1
   TO2000:NEXT:GOTO20
70 CLS:ON R GOSUB1000,2000,3000,4000
80 GOTO20
1000 SL=1:GOSUB5000
1010 SREADD,T,S,A$(0),A$(1)
1020 RETURN
2000 F=1:H=1:CLS:PRINT"aSCII OR hEX
   LISTING ?"
2010 R$=INKEY$:IFR$<>"A"AND
   R$<>"H" THEN2010
2020 AS=0:IFR$="A" THENAS=1
2030 IFF=0 THEN2050
2050 PK=96:CP=1535:IFAS=1 GOSUB
   2320 ELSEGOSUB2280
2050 POKECP,PK:CP=1024+Y*32+X*3:
   PK=PEEK(CP):POKECP,239
2060 PRINT@321,"TOP BYTE=";H
2070 R$=INKEY$:IFR$="" THEN2070
2080 R=INSTR(C$,R$):IFR=0 THEN2070
2090 F=0:ON R GOTO 2100,2110,2120,
   2130,2140,2150,2160,2170
2100 Y=Y-1:GOTO2210
2110 Y=Y+1:GOTO2210
2120 X=X-1:GOTO2210
2130 X=X+1:GOTO2210
2140 AS=1:GOTO2040
2150 AS=0:GOTO2040
2160 RETURN
2170 PRINT@384,"INPUT NEW CONTENTS
   (HEX)□";:INPUTH$
2180 V$=CHR$(VAL("&H"+H$)):P=H+
   Y*11+X
2190 MID$(A$(P/128),P+128*(P>128),
   1)=V$
2200 F=1:GOTO2030
2210 IFY<0 THENH=H-44:Y=0:F=1
2220 IFY>7 THENH=H+44:Y=7:F=1
```



```
2230 IFX<Ø THENX=1Ø:Y=Y−1:IFY<Ø
   THENH=H−11:Y=Ø:F=1
2240 IFX>1Ø THENX=Ø:Y=Y+1:IFY>7
   THENY=7:H=H+11:F=1
2250 IFH=−1ØORH=−43 THENH=1:
   F=Ø:ELSEIFH<1 THENH=1:F=1
2260 IFH=179ORH=212 THENH=168:
   F=Ø ELSEIFH>168 THENH=168:F=1
2270 GOTO2Ø3Ø
2280 CLS:FORJ=H TOH+87 STEP11:FOR
   T=ØTO1Ø
2290 PRINTRIGHT$("Ø"+HEX$(ASC
   (MID$(A$(J/128),J+T+128*
   ((J+T)>128)))),2);"□";
2300 NEXT:PRINTCHR$(8);:NEXT
2310 RETURN
2320 CLS:FORJ=H TOH+87 STEP11:FOR
   T=ØTO1Ø
2330 G=ASC(MID$(A$(J/128),J+T+128*
   ((J+T)>128))):IFG<32 THEN235Ø
2340 PRINT"□";CHR$(G);"□";:GOTO236Ø
2350 PRINTLEFT$("Ø"+HEX$(G),2);"□";
2360 NEXT:PRINTCHR$(8);:NEXT:RETURN
3000 CLS:PRINT"SAVE TO SAME SECTOR
   (Y/N) ?"
3010 R$=INKEY$:IFR$<>"Y"AND
   R$<>"N" THEN3Ø1Ø
3020 IFR$="Y" THEN3Ø4Ø
3030 CLS:GOSUB5ØØØ
3040 PRINT:PRINT"ARE YOU SURE (Y/N) ?"
3050 R$=INKEY$:IFR$<>"Y"
   ANDR$<>"N" THEN3Ø5Ø
3060 IF R$="N" THENRETURN
3070 SWRITED,T,S,A$(Ø),A$(1)
3080 RETURN
4000 GOSUB5Ø5Ø
4010 PRINT#PR,TAB(14);"START□□NO."
4020 PRINT#PR,"□□NAME□□TYPE□
   TR□□□SC□SECS□LEN"
4030 FORJ=ØTO15:SREAD1,16,J+3,D$(Ø),
   D$(1)
4040 FORK=1TO25Ø STEP25
4050 GOSUB6ØØØ
4060 IFASC(V$)<>Ø ANDASC(V$)<>2
   THEN412Ø
4070 PRINT#PR,MID$(V$,2,8);TAB(8);".";
   MID$(V$,1Ø,3);
4080 TS=−1:FORP=13TO22 STEP3:
   V=256*ASC(MID$(V$,P))+ASC(MID$
   (V$,P+1)):EB=ASC(MID$(V$,P+2)):
   TS=TS+EB:IF EB=Ø THEN411Ø
4090 IFP<>13 THENPRINT#PR
4100 PRINT#PR,TAB(12);INT(V/18);TAB(16);
   1+V−18*INT(V/18);TAB(2Ø);ASC(MID$
   (V$,P+2));
4110 NEXTP:PRINT#PR,TAB(24);256*TS+
   ASC(MID$(V$,25))
412ØNEXTK,J:R$=INKEY$:IFPR=−2 THEN
   414Ø
4130 R$=INKEY$:IFR$="" THEN413Ø
4140 RETURN
5000 INPUT"TRACK NUMBER (Ø−39)□";T
5010 INPUT"SECTOR NUMBER
   (1−18)□";S
5020 INPUT"DRIVE NUMBER (1−4)□";D
5030 IFD>4ORD<1ORT>39ORT<ØOR
   S>18ORS<1 THEN5ØØØ
5040 RETURN
5050 PR=Ø:IF(PEEK(65314)AND1)=1 THEN
   RETURN
5060 PRINT"OUTPUT TO PRINTER (Y/N) ?"
5070 R$=INKEY$:IFR$<>"Y"AND
   R$<>"N" THEN5Ø7Ø
5080 IFR$="Y" THENPR=−2
5090 RETURN
6000 V$=MID$(D$(K/128),K+128*
   (K>128),25):IFLEN(V$)<25 THENV$=
   V$+MID$(D$(1+K/128),1,25−LEN(V$))
6010 RETURN
```

USING THE DRAGON DMON

Type in or LOAD the program then insert the disk you intend to work on. Use an unimportant disk while you are practising. Type RUN and you'll see a menu offering four options: Load sector, View/edit sector, Save sector, Catalogue. Press C first and you will see a detailed catalogue of all your files. The list shows the file name, type, start track and sector, number of sectors and the length in bytes. If a file consists of several linked sections then the start track and sector of each section is shown. If you have a printer it is worth taking a copy to keep with your disk.

Now type L to load a sector. Try the directory first, track 16, sector 3. Type V to view. Press A or H at any time to see the listing in either ASCII or hex. You should be able to relate the numbers or letters to the directory map shown earlier.

Use the arrow keys to move the cursor to any byte you wish to change then type in the new value (in hex). Press space to enter the number. Only part of the sector is shown on the screen. To see the rest, move the cursor to the bottom line and the screen will scroll up. The current number of the byte in the top left corner is continuously displayed.

When you've finished editing the sector you can save it back on the disk by returning to the menu and pressing S.

When a file is deleted on the Dragon, the program descriptor byte in front of the file name is changed to 81, but the file name is not deleted. Using the disk monitor to look at the directory you can easily find this byte and change its value to ØØ, for a valid file. You have to change this byte in both track 16 and 20. If you now press BREAK and type DIR you'll see that the file name has reappeared, and you can load the program.

The best thing to do now is to SAVE this program on another disk and then delete the file on the old disk once more. This is because, although you have reinstated the file name, you have not reset the bits relating to that file in sectors 1 and 2 of the directory. It is possible to set these bits, but it is extremely difficult to find the correct ones, and if you make a mistake you could corrupt other files on your disk.

MUSIC MICROS, AND MIDI

Micro music is only just starting to open up. The introduction of MIDI-based equipment allows you to link computers to music synthesizers and run one off the other . . .

Sound has become one of the features most people expect to find on a home microcomputer—some people may even be swayed towards buying a particular computer because of its sound capabilities. In addition to the music you can make on your computer, a way of connecting many home microcomputers to synthesizers and other kinds of musical instruments is now being introduced and becoming freely available. The standard is called MIDI—Musical Instrument Digital Interface—and opens up a whole new range of possible uses for your computer.

SOUND FROM MICROCOMPUTERS

Sound from microcomputers has come a long way from the buzzes and bleeps produced by the earliest models with sound, towards being able to play music and produce sound effects. Of the computers covered in *INPUT*, the Spectrum offers very simple sound, using its BEEP command and the Dragon can be programmed to play tunes via its PLAY command. On the Spectrum and the Dragon, the notes always sound the same—a very electronic-sounding pure tone. The computers also cannot produce more than one note at a time from BASIC.

With the Commodore 64 and the BBC it's a slightly different story. They have what are amongst the most sophisticated sound facilities to be found on microcomputers, both computers having their own dedicated sound chip, which offers a far greater range of musical possibilities. Both the computers have three musical channels, or voices, (the BBC has a noise channel, too) which can be played either on their own or together (single notes or chords). You aren't stuck with a single note 'quality' either—you can shape the sound using envelopes (see pages 1138 to 1144).

The Acorn Electron is a stripped-down version of the BBC, with only one music channel or one noise channel, although retaining all of the BBC's other sound facilities.

If you have typed in any of the sound programs in INPUT (such as those on pages 701 to 707 or pages 985 to 991) you will know what is possible from your computer. Even the most sophisticated sounds that can be produced by the most sophisticated computers are not up to performance or recording standards, not to mention the drawbacks of trying to play music on a QWERTY keyboard. Even at its best, the micro falls some way short of a purpose-built musical instrument.

MUSICAL INSTRUMENTS

The story of musical instrument development in the last few decades closely parallels that of calculating machines. Traditionally, instruments were mechanical—skins being hit, strings being plucked or bowed, and so on. Gradually, the increasing need for greater volume in live performances, and the demands of recording, led to instruments such as guitars and pianos being electrified—and finally, over the past few years, purely electronic musical instruments, such as synthesizers, have appeared. Just as calculating machines have evolved from the mechanical abacus to the modern microcomputer incorporating digital electronic technology, the latest musical instruments are filled with microchips.

Modern synthesizers are extremely sophisticated devices. Instead of the limited number of notes which can be played on a computer, and the limited (or non-existent) enveloping facilities, you'll be faced with a bewildering array of possibilities. Typical, medium-priced synthesizers allow you to play chords of up to eight notes on a proper keyboard. Almost all machines offer an array of preset sounds, so if you want the sound of a piano, or a violin, you merely have to press the correct button. Presets aren't the whole story though, you can twiddle to your heart's content to produce almost any sound you desire—the ad-man's dream of the synthesizer that can be a whole orchestra in your front room isn't here yet, but is probably lurking in the wings somewhere.

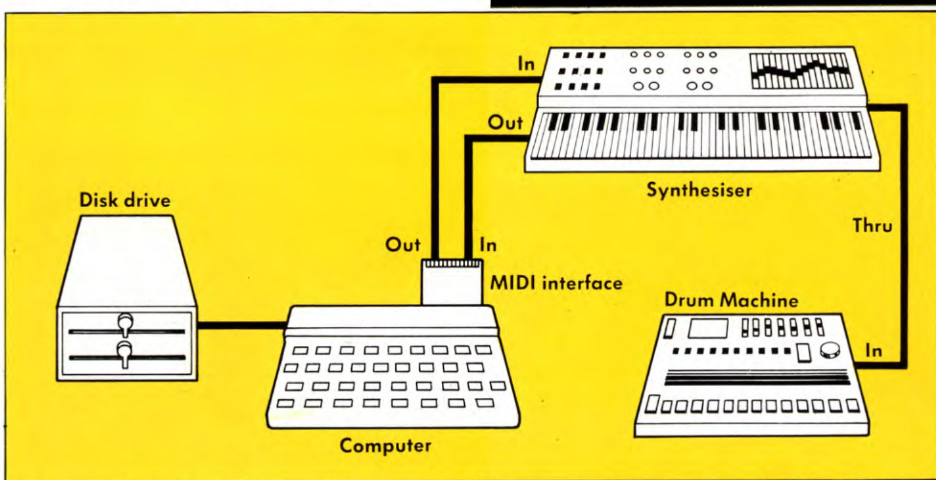
When synthesizers are mentioned, most people immediately think of the keyboard instruments, which are by far the most common type of synthesizer. But since the heart of the synthesizer is really just a box of electronics for producing sounds, which can be triggered by some kind of signal, there is

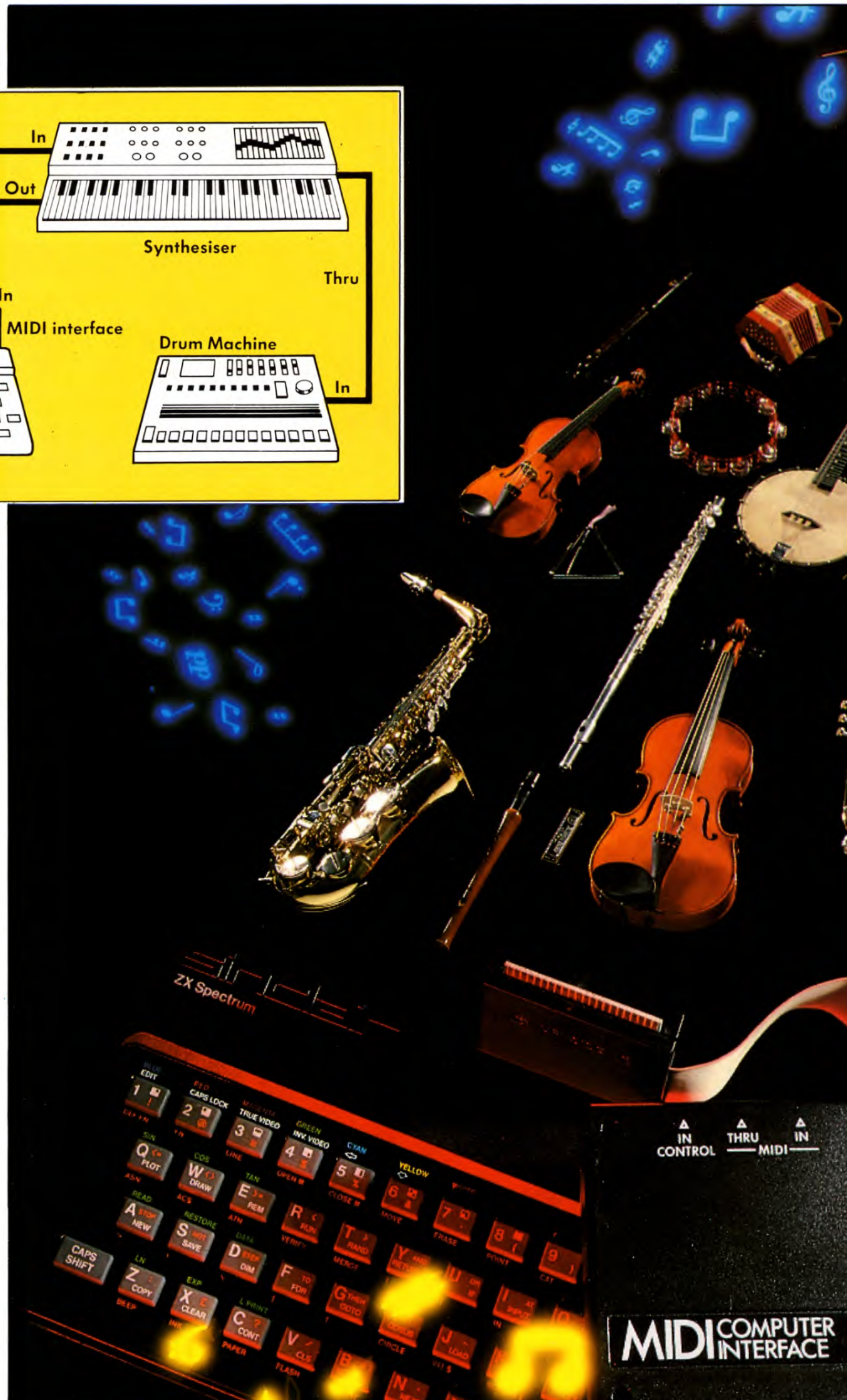


- SYNTHESIZERS
- COMPUTER SOUND
- MUSICAL INSTRUMENTS
- KEYBOARDS
- DRUM MACHINES
- MIDI INTERFACE
- CONNECTING COMPUTERS TO SYNTHESIZERS
- SOUND CAPABILITIES
- SOFTWARE



MIDI linking a computer, a drum machine and a synthesizer. Musical information can be stored on disk





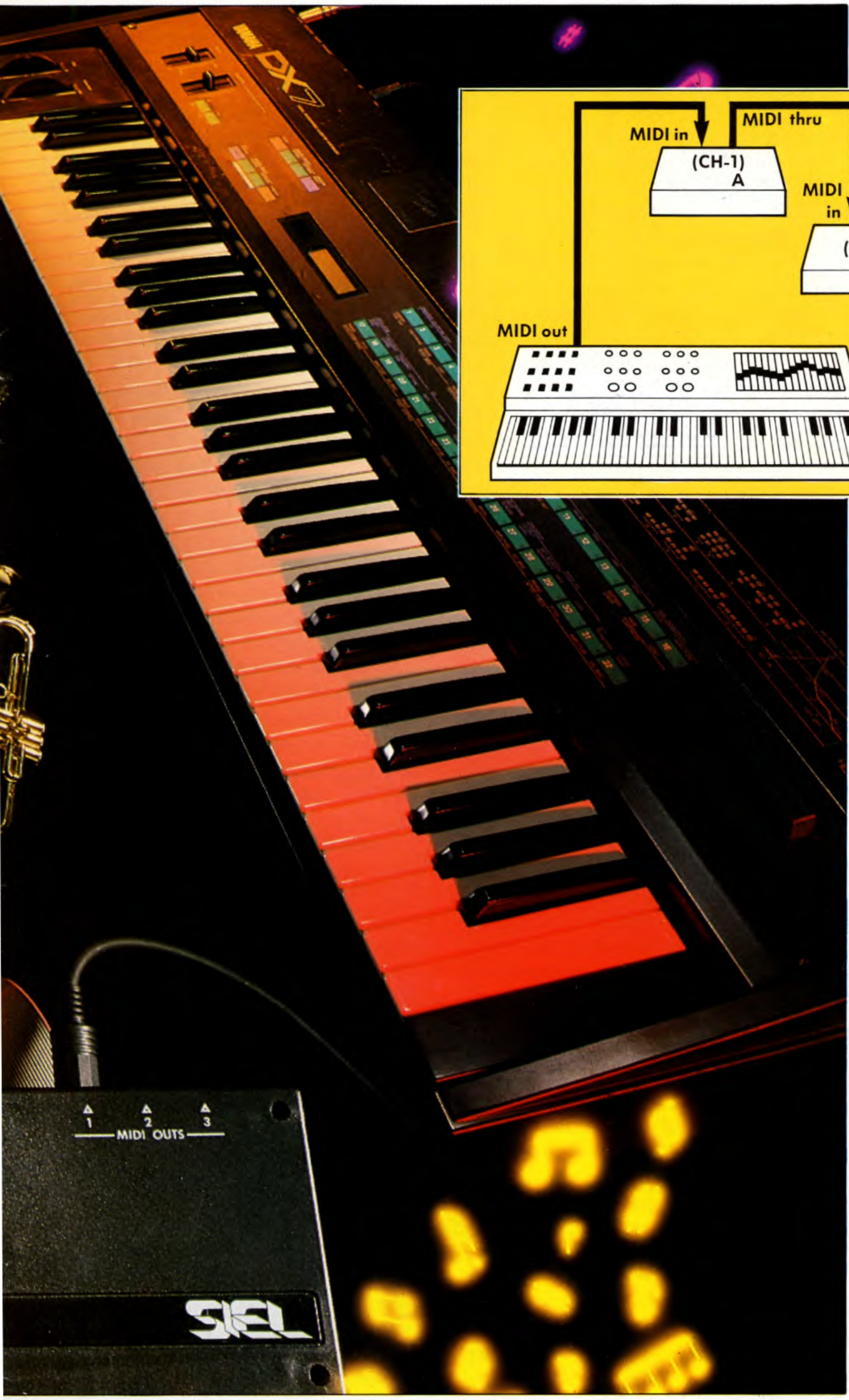
no reason, in theory, why any kind of instrument cannot be used for triggering the electronics. In practice, it's a slightly different story. For various technical reasons, the keyboard remains the most popular kind of synthesizer instrument, although you can buy guitar synthesizers, which are played exactly like a guitar, but sound however you wish. There are also drum synthesizers—as distinct from synthetic drum machines—which are triggered by striking a series of pads.

Drum machines, on the other hand, are pre-programmed to provide a rhythmic backing without the intervention of a performer. But these, too, come within the range of synthesizers. Until recently, drum machines had a very characteristic sound, so any record made with a machine instead of a drummer was instantly recognizable. As technology advances, it's getting rather more difficult to tell. Most drum machines offer some preset rhythms and also have memory facilities which allow you to create and store your own rhythm pattern.

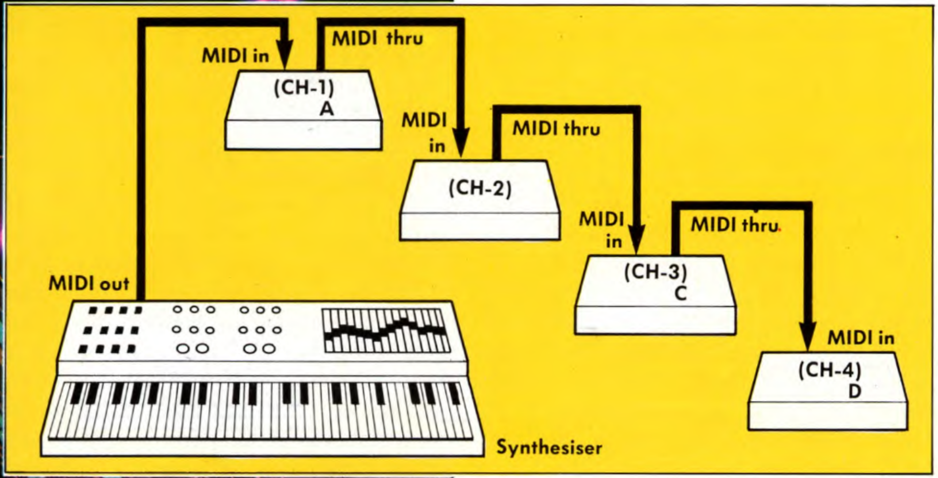
It's this last facility of drum machines which points the way towards a real musical breakthrough. Until recently, musical ability has always been dependent on manual dexterity—being able to move one's fingers quickly and fluently over a keyboard, or being able to hit a drum accurately and on time. With wind instruments, the tricky skill of breath control comes into it, too. The advent of the programmable musical instrument changes all that.

The programmable drum machine isn't a substitute for musical talent. You still need to be able to understand rhythm, and to 'hear' the desired effect in your mind. What the machine does do is to free musical talent from a dependence on manual dexterity—and in this case from a need to purchase an expensive and bulky drum kit.

The keyboard synthesizer already has the facility to sound like virtually any instrument you choose. So if you add an ability to program its performance, a whole world of music opens up even to those people whose fingers are all thumbs. Enter MIDI, a system which allows you to add programming to the synthesizer.



A MIDI bus can contain up to 16 channels of information, controlling as many as 16 instruments



Now that both computers and musical instruments are employing the same kind of technology it's relatively easy to send information from a computer to a musical instrument, or vice versa—what MIDI is all about.

WHAT IS MIDI?

MIDI is a standard just like a Centronics or RS 232 interface which you may have come across in connection with peripherals such as printers or modems. In this case, although MIDI is used exclusively in the world of music, the role of the interface is exactly the same—to transfer standard-formatted information between one place and another. Its main role as a standard is to make sure that the information is transmitted in such a way that any MIDI-compatible piece of equipment will understand the information received.

There have been previous attempts to impose a standard for communication between musical instruments which have failed to become generally accepted. The story seems different for MIDI—all of the synthesizers and drum machines being produced by two of the world's leading keyboard manufacturers, Yamaha and Roland, adhere to the standard, and now a range of computer equipment compatible with the standard is arriving. The standard seems set to be universally adopted by all manufacturers both of electronic musical equipment and of the computer equipment which can be used with them.

Each piece of MIDI-compatible equipment has three five-pin DIN sockets. These are labelled 'IN', 'OUT' and 'THRU' (some older MIDI equipment may not have 'THRU'). 'IN' allows the equipment to receive MIDI signals from another piece of MIDI equipment. 'OUT' is simply the reverse, allowing one piece of MIDI equipment to send out MIDI signals to another piece of MIDI equipment. 'THRU' sends a direct copy of the incoming information on to another piece of MIDI equipment. This means you can drive several pieces of equipment at the same time by connecting together via the 'THRU' sockets. So equipment which does not have 'THRU' is much more limited.

MIDI allows up to 16 separate channels of information to be transmitted simultaneously. Each channel allows the musician to control a separate instrument, but the information co-exists in the same wire. Each piece of equipment 'tunes in' to the information being sent to it, a little like a television receiver tunes in to a particular television channel.

HOW DO I USE MIDI?

MIDI has been around since 1982, although it has only just been brought to the attention of home computer owners. Musicians have been using MIDI to trigger one instrument from another. For example, two synthesizers can be set to produce different sounds, but be played simultaneously from one keyboard, by connecting the two instruments together via MIDI and playing the keyboard of one of the two instruments.

MIDI also allows a musician to connect a drum machine to a keyboard and synchronize a rhythm track with the melody. Other possibilities include connecting a sequencer. A sequencer is a device which remembers what has been played and allows it to be played back. There are two types—real-time and step-time. A real-time sequencer plays back exactly what the musician has played, whereas a step-time sequencer literally steps through the tune, with the musician playing each note in turn, filling in individual slices of time, until the tune is completed.

MIDI AND HOME COMPUTERS

Along with the launch of computers using the MSX standard has come publicity about the relationship between MIDI and home computers. Yamaha have introduced the CX5M music computer, an MSX computer with a built-in synthesizer. Adding a piano-type keyboard to the computer gives the owner a fully fledged synthesizer. The machine opens up all sorts of possibilities to musicians—music can be composed on a monitor screen, or the computer can be used as a sequencer without extra hardware.

This computer costs considerably more than a BBC, but you can get a similar set-up if you own a Spectrum, Commodore 64, or BBC by using your computer connected to a MIDI synthesizer. You'll need a MIDI interface box to plug into your computer, a connecting lead, and some software.

Costing less than a Spectrum, the interface box will allow you to connect your computer to any piece of MIDI-compatible equipment. At present, with the price of MIDI-compatible synthesizers starting at well in excess of a BBC, this is an expensive way for home computer users to extend their music making capabilities, but like printers, colour monitors and disk drives, the prices of instruments can be predicted to fall. In the near future a synthesizer will probably be comparable in price to the home micros that can be used to control them.

But even before prices fall, owners of MIDI-compatible musical equipment will find that a home computer and interface is a very attractive proposition. With suitable software a whole range of possibilities is opened up, and any number of dedicated add-ons can be imitated, at a fraction of the cost.

The built-in sound capabilities of the chosen home computer are not used at all when connected to MIDI—the sound is always generated by the synthesizer or drum machine—so there isn't really any point in purchasing an expensive microcomputer for use specifically with MIDI. It is interesting to realise that there is no real advantage in using expensive business computers over home computers. Even the extra memory offered by a business machine is largely superfluous, as the standard memory size of a home computer generally offers far more storage than any dedicated sequencer, for example. The message is that any computer that can have a MIDI interface attached is just as good as another, although it must be said that you may well find a 16K Spectrum slightly limited in its storage capacity.

It's worth noting, too, that the sound quality available is not limited by the recording medium. Because the sound is stored digitally, it should be comparable with a medium like Compact Disc rather than, say, tape where all sorts of unwanted noise may be introduced. In other words, what comes out of a MIDI system is exactly what went in.

MIDI SOFTWARE

Once you have your computer hitched up via MIDI to your musical instrument, you'll need some software to make it all work. At present, the range is still restricted and comparable in price to some of the business software available for home computers. The situation will change as more and more people want to use MIDI.

Nonetheless, even within the restricted software range, there is software which will enable you to duplicate sequencers, compose multitrack music, and edit your tunes. Exactly what is on offer varies from manufacturer to manufacturer, and from computer to computer.

Although you may not be able to play a note on any musical instrument, you'll find that you can play music by composing on your computer's monitor, and sending the information to the musical instrument to be played. The composition can also be stored on disk or tape for playing back or alteration at a later date. It has been predicted that some sheet music will be available on MIDI-coded EPROMs, so you can have either whole pieces of music which can be played back a little like a record or tape, or you could play along with an EPROM containing a backing track.

A typical MIDI software package is the music composer program which allows you to build up your piece on screen in much the same way as you would write it on paper—adding notes to a musical staff. Full replay and editing facilities allow you to check your progress and make alterations to the music on screen as you go.

But it doesn't stop there, as good software should put all the features of the synthesizer at your disposal. You can control as many voices at once as there are voices on the synthesizer—a typical good-quality polyphonic keyboard synthesizer may be able to play as many as 16 notes together. You can select from the synthesizer's range of preset voices or blend new notes. If your synthesizer has split keyboard capability, you can even have two different instruments playing together—a melody and backing, perhaps. There are generally three types of information which you can send through the MIDI—notes, program changes and pitchblend.

At present, there is a standard set of MIDI codes which work with any MIDI-compatible synthesizer. But these tend only to control the most basic functions available. Special features are accessed by special, extended code systems—and these usually vary from instrument to instrument. As a result, a complex, widely orchestrated composition may call for familiarity with a large number of MIDI codes, although this may well simplify considerably in future.

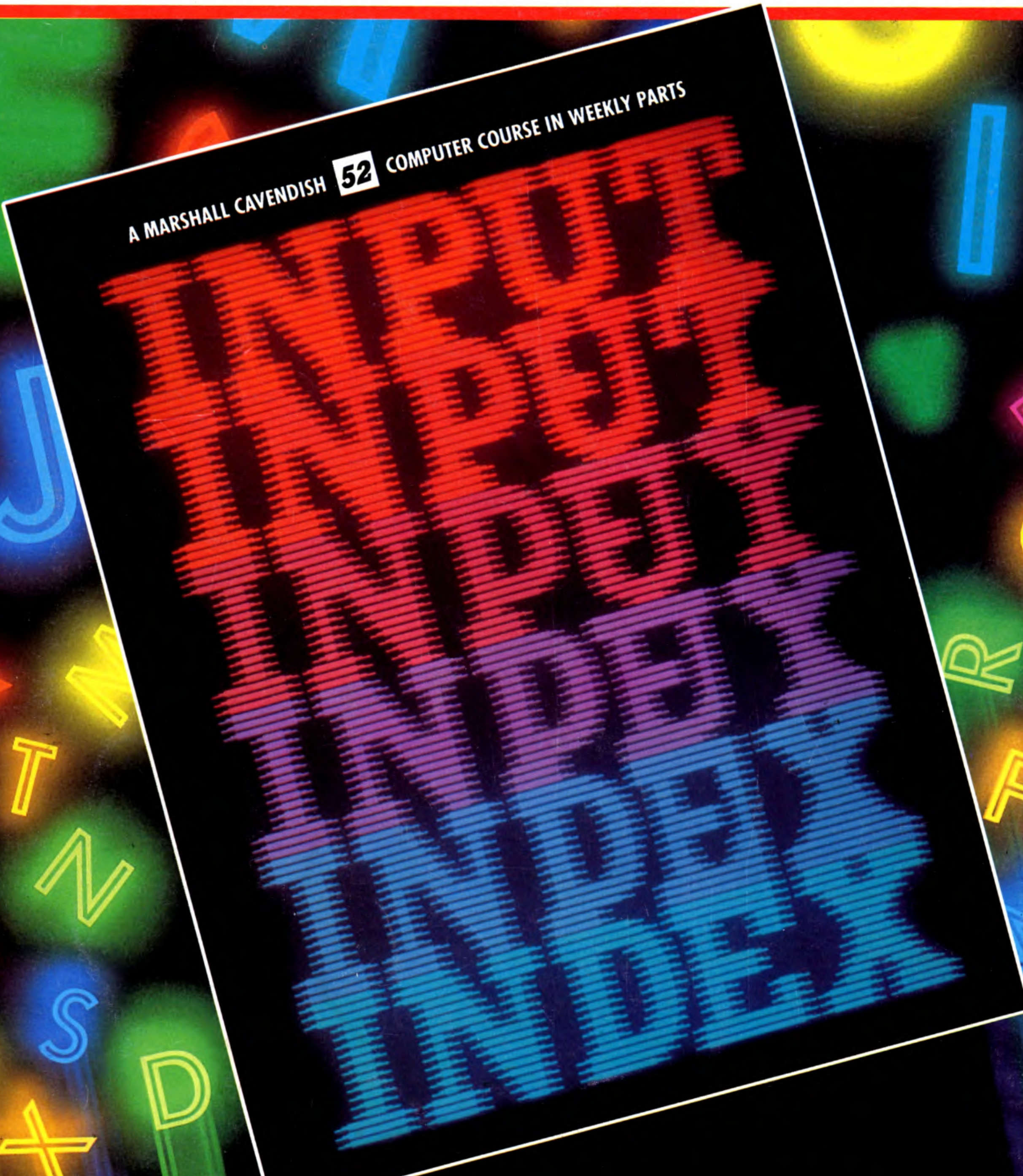
There's no reason why you should be discouraged from using MIDI, because you are a computer programmer rather than a musician. In some ways you have a positive advantage. If you can program in machine code there's nothing to stop you writing your own MIDI software, tailored to your own needs, and save the expense of buying commercial software.

MIDI seems to offer the musician and non-musician alike many new possibilities. As prices fall, musical instruments seem destined to find their way into far more homes—perhaps it'll be back to the old sing-song evenings round the synthesizer! And you will have the opportunity of being really creative with your computer.

CUMULATIVE INDEX

An interim index will be published each week. There will be a complete index in the last issue of *INPUT*.

COMING IN ISSUE 52...



The final part of INPUT forms a complete index to your collection. There's a full list of all the articles that make up the course, plus a handy A-Z reference guide to all the subjects covered.